



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

2014-09

Digitizing consumption across the operational spectrum

Branham, Andrew R.

Monterey, California: Naval Postgraduate School

<http://hdl.handle.net/10945/43880>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DIGITIZING CONSUMPTION ACROSS THE OPERATIONAL SPECTRUM

by

Andrew R. Branham

September 2014

Thesis Advisor:
Second Reader:

John Gibson
Thomas Otani

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2014	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE DIGITIZING CONSUMPTION ACROSS THE OPERATIONAL SPECTRUM			5. FUNDING NUMBERS	
6. AUTHOR(S) Andrew R. Branham				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) <p>U.S. Marine Corps logisticians and operational planners must simultaneously plan for the sustainment of current operations while planning for future operations. Currently, this process is hindered by the manual correlation of force consumption data from electronic and hardcopy documents.</p> <p>In order to refine this process, this thesis presents a process for converting, analyzing, and storing these documents in an electronic format. In order to aid in the conversion process, three optical character recognition (OCR) applications are compared: an open-source and freely-available online application, Microsoft OneNote®, and Nuance OmniPage®. Two data extraction programs were created and compared to assess the feasibility of automating the analysis phase. The first program concentrated on automated analysis with user review at the end. The second program concentrated on continual user interaction throughout the entire process.</p> <p>The results of these comparisons advocate the use of professional-grade OCR software such as OmniPage® to create a standard file that can be accepted as an input by a data extraction program. Based on the consumption documents reviewed by this thesis, a manual data extraction program is advised to create a universal output format for later use in an appropriate data storage method.</p>				
14. SUBJECT TERMS Logistics, relational database, non-relational database, NoSQL, optical character recognition, walkthrough analysis, automated analysis, data conversion, data extraction, document repository, consumption data			15. NUMBER OF PAGES 111	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DIGITIZING CONSUMPTION ACROSS THE OPERATIONAL SPECTRUM

Andrew R. Branham
Lieutenant, United States Navy
B.A., Columbia College, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
September 2014**

Author: Andrew R. Branham

Approved by: John Gibson
Thesis Advisor

Thomas Otani
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

U.S. Marine Corps logisticians and operational planners must simultaneously plan for the sustainment of current operations while planning for future operations. Currently, this process is hindered by the manual correlation of force consumption data from electronic and hardcopy documents.

In order to refine this process, this thesis presents a process for converting, analyzing, and storing these documents in an electronic format. In order to aid in the conversion process, three optical character recognition (OCR) applications are compared: an open-source and freely-available online application, Microsoft OneNote®, and Nuance OmniPage®. Two data extraction programs were created and compared to assess the feasibility of automating the analysis phase. The first program concentrated on automated analysis with user review at the end. The second program concentrated on continual user interaction throughout the entire process.

The results of these comparisons advocate the use of professional-grade OCR software such as OmniPage® to create a standard file that can be accepted as an input by a data extraction program. Based on the consumption documents reviewed by this thesis, a manual data extraction program is advised to create a universal output format for later use in an appropriate data storage method.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	MOTIVATION	1
	1. The Current Methodology of Manual Correlation.....	2
	<i>a. Location.....</i>	<i>2</i>
	<i>b. Retrieval.....</i>	<i>2</i>
	<i>c. Analyze</i>	<i>4</i>
	<i>d. Utilize.....</i>	<i>4</i>
	2. Present-day Solutions	5
B.	PURPOSE OF STUDY.....	7
C.	PRO/CON ANALYSIS OF AUTOMATION.....	9
D.	ORGANIZATION OF THESIS	9
II.	BACKGROUND STUDY.....	11
A.	WHAT IS OPTICAL CHARACTER RECOGNITION?.....	11
	1. OCR Software Approaches.....	11
	<i>a. Free-form OCR.....</i>	<i>13</i>
	<i>b. Template-based OCR.....</i>	<i>13</i>
	2. Increasing OCR Accuracy	16
	3. OCR Summary.....	18
B.	METHODS FOR STORING CONSUMPTION DATA	18
	1. Traditional Database	18
	<i>a. Relational Database Model.....</i>	<i>18</i>
	<i>b. Non-relational Database Model</i>	<i>20</i>
	<i>c. Database Architecture.....</i>	<i>23</i>
	2. Document Repository	25
	3. Data Storage Summary	26
III.	ANALYSIS PROGRAMS.....	27
A.	PROGRAM CREATION.....	27
	1. Language Selection	27
	2. Design and Functionality Specification.....	27
	<i>a. Data Import (1.1)—Open the Input File.....</i>	<i>28</i>
	<i>b. Read in Inputs and Store, Close Input File (1.2 and 1.3)</i>	<i>30</i>
	<i>c. Analyze, Verify, and Correct Inputs (2.1–2.3)</i>	<i>32</i>
	<i>d. Export Analyzed Data (3.1 and 3.2)</i>	<i>37</i>
IV.	FIELD DEMONSTRATION, TESTING, AND RESULTS	39
A.	OCR TESTING AND COMPARISON	39
	1. Testing Environment	39
	2. OCR Applications	45
	3. Online, Open-Source OCR.....	48
	4. Microsoft OneNote® OCR.....	55
	5. Nuance OmniPage® OCR.....	61
	6. OCR Summary.....	70

B.	PROGRAM DEMONSTRATION	70
1.	Automated Program	71
a.	<i>File Input and Closing</i>	<i>71</i>
b.	<i>Detect and Extract Document Information</i>	<i>73</i>
c.	<i>Detect and Extract Table Information</i>	<i>75</i>
d.	<i>File Output</i>	<i>79</i>
e.	<i>Phase Two Test.....</i>	<i>80</i>
f.	<i>Automated Program Summary.....</i>	<i>81</i>
2.	Walkthrough Programs.....	82
a.	<i>Line-by-line Program.....</i>	<i>83</i>
b.	<i>Page-by-page Program</i>	<i>84</i>
3.	Program Summary	85
V.	CONCLUSIONS	87
A.	LESSONS LEARNED	87
1.	Choosing OCR Application.....	87
2.	Choosing Application Approach	87
B.	RECOMMENDATIONS.....	89
C.	FUTURE RESEARCH.....	89
D.	SUMMARY	90
	LIST OF REFERENCES	91
	INITIAL DISTRIBUTION LIST	93

LIST OF FIGURES

Figure 1.	Consumption Data Correlation Process	1
Figure 2.	Infantry-Heavy Threat Combat Planning Factors Table (from [2]).....	2
Figure 3.	Class V(W) FY-06 Planning Factors Consumption Data (from [3]).....	4
Figure 4.	Fuel Planning Worksheet in Microsoft Excel® (from [3]).....	5
Figure 5.	Spreadsheet utilized by II Marine Expeditionary Force (from [3])	6
Figure 6.	Consumption Data Correlation Process (refined)	7
Figure 7.	Usage Table—Pre-OCR (from [2])	12
Figure 8.	Usage Table—Post-OCR (after [2])	12
Figure 9.	Blank Check for Template-based OCR Discussion.....	14
Figure 10.	Template-based OCR Process of a Check	16
Figure 11.	Consumption Data Element in Relational Form	19
Figure 12.	Relational Database Query and Result	20
Figure 13.	NoSQL (key, value) Dictionary Example.....	21
Figure 14.	Java-implemented Dictionary and Query: Result	22
Figure 15.	Global Database Architecture	23
Figure 16.	Database Location Hardware Instance.....	24
Figure 17.	DOD Instructions in Circulation (from [4]).....	25
Figure 18.	Application Flow	28
Figure 19.	Open Input File Routine with Line Counting Logic.....	29
Figure 20.	Read-in and Storage of Inputs.....	31
Figure 21.	Walkthrough Analysis Inputs—Consumptiondata.txt	33
Figure 22.	Program to Execute Walkthrough Analysis.....	34
Figure 23.	Program to Execute Automated Analysis	36
Figure 24.	EX-1, Marine Corps Order Front Page (from [2]).....	41
Figure 25.	EX-2, Usage Table in Profile View (from [2])	42
Figure 26.	EX-3, Usage Table as an Image (from [2]).....	43
Figure 27.	EX-4, Usage Table as a Table (from [2])	44
Figure 28.	http://www.onlineocr.net : Main Page and Features (after [9])	49
Figure 29.	Post-OCR Results of EX-1 using onlineocr.net (after [2])	50
Figure 30.	Post-OCR Results of EX-2 using onlineocr.net (after [2])	51
Figure 31.	Post-OCR Results of EX-3 using onlineocr.net (after [2])	52
Figure 32.	Post-OCR Results of EX-4 using onlineocr.net (after [2])	53
Figure 33.	Post-OCR Results of EX-1 using OneNote® (after [2])	55
Figure 34.	Post-OCR Results of EX-1 using OneNote®: Second Pass (after [2])	56
Figure 35.	Post-OCR Results of EX-2 using OneNote® (after [2])	57
Figure 36.	Post-OCR Results of EX-3 using OneNote® (after [2])	58
Figure 37.	Post-OCR Results of EX-4 using OneNote® (after [2])	59
Figure 38.	OmniPage Start Screen	61
Figure 39.	OmniPage® Workspace.....	62
Figure 40.	OmniPage® Workspace post-OCR	63
Figure 41.	OmniPage® Proofreader.....	64
Figure 42.	Post-OCR Results of EX-1 using OmniPage® (after [2])	65

Figure 43.	Post-OCR Results of EX-2 using OmniPage® (after [2])	66
Figure 44.	Post-OCR Results of EX-3 using OmniPage® (after [2])	67
Figure 45.	Post-OCR Results of EX-4 using OmniPage® (after [2])	68
Figure 46.	Automated Program: File Open and Close	72
Figure 47.	Automated Program: Detect and Extract Document Info	73
Figure 48.	Detect and Extract Document Info Output	74
Figure 49.	Automated Program: Handle Data Structures (v1.0)	77
Figure 50.	Handle Data Structures Output (v1.0)	78
Figure 51.	Automated Program: Handle Data Structures (v2.0)	78
Figure 52.	Automated Program: File Output	79
Figure 53.	Automated Program: Handle Consolidated Input File	81
Figure 54.	Line-by-line Program (coding)	83
Figure 55.	Line-by-line Program (running)	83
Figure 56.	Page-by-page Program (coding)	84
Figure 57.	Page-by-page Program (running)	85

LIST OF TABLES

Table 1.	OCR Input File Types	46
Table 2.	OCR Output File Types	47
Table 3.	Online, Open-Source OCR Results and Findings.....	54
Table 4.	OneNote® Results and Findings	60
Table 5.	OmniPage® Results and Findings	69
Table 6.	OCR Summary Scores	70

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACE	Aviation Combat Element
ANSI	American National Standards Institute
ARG	Amphibious Readiness Group
BA	basic allowance
CE	command element
COTS	commercial-off-the-shelf
CSSE	Combat Service Support Element
CSV	Comma Separated Values
DOD	Department of Defense
DODIC	Department of Defense Identification Code
FMF	Fleet Marine Force
FOB	forward operating base
GCE	Ground Combat Element
GIGO	garbage in, garbage out
HCI	Human Computer Interaction
HTML	Hypertext Markup Language
I/O	input/output
JSON	JavaScript Object Notation
MAGTF	Marine Air-Ground Task Force
MCO	Marine Corps Order
MEF	Marine Expeditionary Force
MEU	Marine Expeditionary Group
MCP	Marine Corps Planning Factors
MICR	magnetic ink character recognition
MRE	Meal Ready to Eat
NOSQL	Not only SQL
OCR	optical character recognition
PDF	Portable Document Format
QS	query size
RS	result size

SQL	Structured Query Language
STDOUT	Standard Output
TPD	transactions per day
USMC	United States Marine Corps
USN	United States Navy
XML	Extensible Markup Language

ACKNOWLEDGMENTS

I would like to thank my wife, Amber, for her love and devotion throughout my time at the Naval Postgraduate School.

I would also like to thank Professor John Gibson and Professor Thomas Otani for their valuable inputs and feedback.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The United States Marine Corps, established in November 1775, has participated in countless operations around the world. Tasked with a multitude of operations ranging from major armed conflicts to humanitarian aid missions, U.S. Marine Corps logisticians and operational planners must continually plan, maintain, and execute acquisition and distribution programs to support approximately 174,000 troops according to [1]. Regardless of the scale of an operation or mission, they must always answer the fundamental logistical question of “how much equipment and supplies do we need for the amount of personnel assigned to the mission?”

In order to answer this question, they must locate the correct reference documents that may exist in electronic and hardcopy formats, retrieve the correct consumption data that often resides in “usage tables,” and analyze these inputs, providing useful planning data for utilization. This cyclic process is depicted in Figure 1. For the purpose of this thesis and from a logistical standpoint, “consumption data” is an all-encompassing term that describes the raw data contained in the logistical planning factor input documents. When a specific example of consumption data is illustrated, it will be presented as a “consumption data element.” The term “usage table” is defined in the next section.

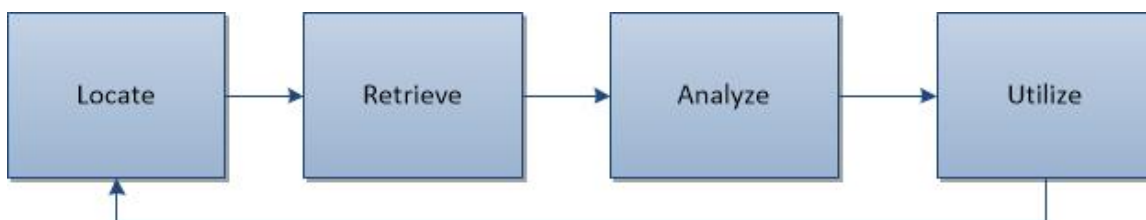


Figure 1. Consumption Data Correlation Process

1. The Current Methodology of Manual Correlation

Under the current methodology, the location, retrieval, and analyzing of consumption data is conducted manually by civilian and military personnel in the logistics or operations field from various locations throughout the world.

a. Location

The first challenge that must be overcome is the collection of the correct reference documents. While some may exist in an electronic format such as a portable document format (PDF), others exist as hardcopy books, field manuals, and orders. Thus, the planner must have access to both electronic and hardcopy resources. Depending on their situation, this may not be possible.

b. Retrieval

The next challenge is locating from these documents the correct consumption data that often reside in tables. These tables, referred to as “usage tables,” represent the standard display format of consumption data elements. Typically, each consumption data element is listed in a table with its corresponding consumption rate. Thus, a usage table is a collection of consumption data elements and their corresponding rate of consumption or allowance. In order to understand the end-user’s use of this tabular data, an illustration of one usage table and its properties is given as an example.

(1) Usage Table. Figure 2 illustrates one instance of a usage table. Each line, composed of four properties (columns), represents one single consumption data element. Note that for the example, some columns are not included for legibility.

Infantry-Heavy Threat Combat Planning Factors Table					
Weapon ID Sequence					
Weapon		Ammunition		GCE RATES	
Weapon ID	Nomenclature	DODIC	Nomenclature	Daily ASSAULT	Daily SUSTAIN
B0471	SQUAD DEMOLITION SET	M032	CHARGE, DEMO BLOCK 1 LB TNT	15.00753	3.39605
B0471	SQUAD DEMOLITION SET	M130	CAP, BLASTING ELECTRIC	18.01945	4.00000

Figure 2. Infantry-Heavy Threat Combat Planning Factors Table (from [2])

This table consists of the following columns:

- **Weapon.** Consists of two fields: Weapon ID and Nomenclature. These two fields are used to uniquely identify the weapon. An expected input for this field is alphanumeric characters.
- **Ammunition.** Consists of two fields: the Department of Defense Identification (DODIC) and Nomenclature. These two fields are used to uniquely identify the ammunition being used by the weapon identified in the “Weapon” column. An expected input for this field is alphanumeric characters.
- **GCE Rates.** This column is used to define the consumption rate of the Ground Combat Element (GCE) component of the Marine Air-Ground Task Force (MAGTF). The GCE is the primary attack element of a MAGTF and is expected to have a higher rate of consumption. An expected input for this field is an integer or floating point (decimal) number. For elements intended for GCE-use only, the “OTHER-THAN GCE RATES” column may be empty. This column is broken down further into three sub-columns:
 - **Daily Assault.** This rate is shown as the number of rounds per day per weapon or individual in the GCE during the assault (intense) phase of combat [2].
 - **Daily Sustain.** This rate is shown as the number of rounds per day per weapon or individual in the GCE during the sustainment phase of combat [2].
 - **Basic Allowance.** This rate indicates the basic allowance (BA) of the ammunition item recommended to be carried within the means normally available to the Fleet Marine Force (FMF) unit embarking and debarking for combat operations [2].
- **Other than GCE Rates.** This column is used to define the consumption rate of the Command Element (CE), Aviation Combat Element (ACE), and Combat Service Support Element (CSSE) of the MAGTF. Overall, this column has the same characteristics of the “GCE RATES” column: daily assault, daily sustain, basic allowance, and the use of integer or floating point numbers. For items intended solely for the CE, ACE, or CSSE, the “GCE RATES” column may be empty.

While not every usage table published by the United States Marine Corps may be an exact replica (data and content) of the one shown in Figure 2, it is reasonably-expected that each of them will follow a similar table layout or uniquely-structured format. Figure 3, for example, represents the same information from Figure 2 as a table from a different reference document.

ENCLOSURE 2: CLASS V(W) FY-06 PLANNING FACTORS							
NOMENCLATURE	TAMCN	DoDIC	NOMENCLATURE	GCE RATES		NON-GCE RATES	
				DAILY ASSAULT	DAILY SUSTAIN	DAILY ASSAULT	DAILY SUSTAIN
Demolition Equipment, Squad	B0471	G900	Grenade, Hand Incendiary Thermite AN-M14	0	0	1.545	1.545
		M032	Charge, Demolition Block TNT 1-Pound	39.057	66.313	7.416	4.574

Figure 3. Class V(W) FY-06 Planning Factors Consumption Data (from [3])

c. Analyze

Having found the correct reference documents and usage tables, the next challenge faced by the planner is to analyze the usage tables, extracting out specific consumption data elements out as necessary. In order to plan for an operation to field 100 Marines for example, they may need to gather 20 consumption data elements from an electronic usage table and 100 consumption data elements from a different hardcopy usage table. In order to keep track of these elements for planning an operation, the most realistic approach is to record them into a single location in a universal format. This is done primarily via electronic means—Microsoft Word®, Microsoft Excel®, text files, etc. Thus, even during the analyzing phase, they must juggle between electronic and hardcopy formats. To complicate matters, the logistician may be stateside or deployed, may or may not have access to the Internet, may or may not have hardcopy usage tables for ready reference, and may not have an extensive planning shop at his/her disposal. Also, the user must have some familiarity with usage tables and a working knowledge of which usage tables contain specific data elements. Since hardcopy usage tables do not have search functionality, inexperienced planning personnel may spend countless hours reading through a usage table document only to find that they had the wrong document. A clear benefit of having electronic usage tables, in the form of a PDF for example, is that the user gains the ability to search through the document using partial or full keyword search ability.

d. Utilize

After the correct information has been located and compiled, military planners and logisticians provide that data to their military commander for planning purposes or use the data to accomplish their main task—equipping and maintaining troops. Thus, depending on the final document, different variations for displaying the data may be

used—databases, spreadsheets, pie charts, etc. While different storage methods are discussed for storing the output of the extraction programs, this stage of the process is outside the focus of this thesis.

2. Present-day Solutions

While no systems have been created to address this specific conundrum, several systems have been created to aid in the planning process. Systems such as the Joint Operations Planning and Execution System and The Marine Air Ground Task Force War Planning System have been used as resources [3]. Some end-users have taken a proactive approach, creating stand-alone systems. Using programs such as Microsoft Excel® and other user-created applications, these users have attempted to provide temporary solutions to the current problem as depicted in Figures 4 and 5.

	A	B	C	D	E	F	G	H	I	J	K
1			Unit Daily	Unit Daily	Normal	Total Daily	Total Daily	Enter	Enter No of	Enter No of	
2			Fuel Rqmt	Fuel Rqmt	Unit	Fuel Rqmt	Fuel Rqmt	Actual Unit	Days in the	Days	
3	Unit Type	T/E No.	Assault	Sustained	Multiple	Assault	Sustained	Multiple	Assault	Sustained	Subtotal
4	CE										
5	MEF HQ	N4601	0	0	1	0	0	1	0	1	0
6	SRIG										
7	H&S Co	N4606	4,523	2,706	1	4,523	2,706	1	7	10	58,721
8	HQ Intell Co	N4607	51	26	1	51	26	1	0	1	26
9	TOPO PIT	N4608	310	208	1	310	208	1	0	1	208
10	SCAMP	N4609	673	389	1	673	389	1	0	1	389
11	FIIU	N4610	34	17	1	34	17	1	0	1	17
12	HUMINT Co	N4613	0	0	1	0	0	1	0	1	0
13		N4614	0	0	1	0	0	1	0	1	0
14		N4615	102	51	1	102	51	1	0	1	51
15	Radio Bn H&S Co	N4637	4,406	3,707	1	4,406	3,707	1	0	1	3,707
16	Radio Co	N4635	1,270	697	2	2,540	1,394	2	0	1	1,394
17	Radio Co	N4636	1,751	903	1	1,751	903	1	0	1	903
18	Force Recon Co	N4618	609	342	1	609	342	1	0	1	342
19	ANGLICO	N4654	1,931	1,361	1	1,931	1,361	1	0	1	1,361
20	Comm Bn HQ Co	N4686	1,106	614	1	1,106	614	1	0	1	614
21	Serv Co	N4683	1,563	1,592	1	1,563	1,592	1	0	1	1,592
22	Gen Spt Comm Co	N4684	4,459	3,435	1	4,459	3,435	1	0	1	3,435
23	Dir Spt Comm Co	N4685	744	610	3	2,232	1,830	3	0	1	1,830
24								CE SUBTOTAL		74,590	
25	GCE										
26	Division										
27	HQ	N1011	0	0	1	0	0	1	0	1	0
28	H&S Co	N1012	3,163	1,725	1	3,163	1,725	1	0	1	1,725
29	Recon Co	N1019	163	102	1	163	102	1	0	1	102
30	Truck Co	N1016	10,403	5,399	1	10,403	5,399	1	0	1	5,399
31	Comm Co	N1015	2,885	1,938	1	2,885	1,938	1	0	1	1,938
32	MP Co	N1014	337	315	1	337	315	1	0	1	315
33	Inf Bn H&S Co	N1111	2,504	1,840	1	2,504	1,840	1	0	1	1,840

Figure 4. Fuel Planning Worksheet in Microsoft Excel® (from [3])

The first three sheets will serve as input sheets. Computations will be made from the inputs that you make. The intent is to enter hard data, such as current personnel strength, and the estimated OPTEMPO of the mission. The BLUE cells are required user inputs. The RED cells are the recommended "book" solution. The YELLOW cells are optional overrides of the book solution.

Day and Type of Mission										
Enter	Your	Day	or	Phase	in	the blue	cell on the	OPTEMPO	sheet!	
ATTACK	DELAY	HASTY DEFENSE	PREPARED DEF	RESERVE	UNCOMMITTED	STATIC	NONE (NO CONS)	ATTACK	DELAY	
Time Factors in Hours										
Cross Country	1	2	3	4	5	6	1	2	3	4
Secondary Roads	1	2	3	4	5	6	1	2	3	4
Tactical Idle	1	2	3	4	5	6	1	2	3	4
Distance in KM	100	200	300	400	500	600	100	200	300	400
Personnel strength 4000										
Tracked Vehicles										
M1	116	M563 SK Tanker	21	ATK HEL		21				
M2/M3	126	HEMTT CGO	22	Utility Helo		22				
MT13 Series	23	HEMTT Fueler	23	CGO HEL		23				
M109FAASY	24	5 Ton Tractor	24							
MLRS	25	PLS	25							
M88	26	HET	26							
WOLVERINE	27	Spare 1	27							
AVLB	28	Spare 2	28							
Spare 1	29	Total number of wheeled Vehicles								
Spare 2	30	800 (Used For Fuel Consumption)								

Figure 5. Spreadsheet utilized by II Marine Expeditionary Force (from [3])

Although these user-created applications are made with good intentions, most suffer from the same deficiencies:

- Not accepted as legitimate applications by the U.S. Marine Corps.
- May contain volatile or malicious code susceptible to attack.
- Not widely-distributable or maintainable.
- May contain invalid and/or obsolete information.

While these solutions are innovative and have some merit, a more stable solution that can be accredited, upgraded, and distributed to all users in the Marine Corps in a variety of environments is necessary.

Another simple and inexpensive solution for today's environment is the use of historical information. When constrained by time or resources, planners may rely on data from previous operations or exercises to plan for an operation. Since many operations may be similar in nature, planning for a new operation may be as simple as changing the total troop count or total vehicle count. This methodology has two main drawbacks: a lack of flexibility and the potential to be trusted as the definitive data without further examination. With the changing environment of the world, new missions may be encountered which lack historical examples. Not only does this lack flexibility, repetitive use of historical data can erode the core competencies of planners and may provide estimates that are clearly inappropriate for the given scenario based on the different factors surrounding the mission.

From a “cradle-to-grave” standpoint, the current methodology of locating, retrieving, analyzing, and utilizing consumption data is a tedious and laborious task. Depending on the type and amount of operations being conducted, this compilation process may become exponentially time-consuming and complicated. Further aggravated by personnel cuts, this process places an undue burden on the planner.

B. PURPOSE OF STUDY

This thesis strives to reduce the burden placed on the planner by answering three main research questions:

- “What are the abilities and limitations of current OCR technologies?”
- “What is the best method for analyzing consumption documents? Automated analysis with review at the end or walkthrough analysis with review throughout the entire process?”

Figure 6 illustrates how answering these questions relates to refining the current process.

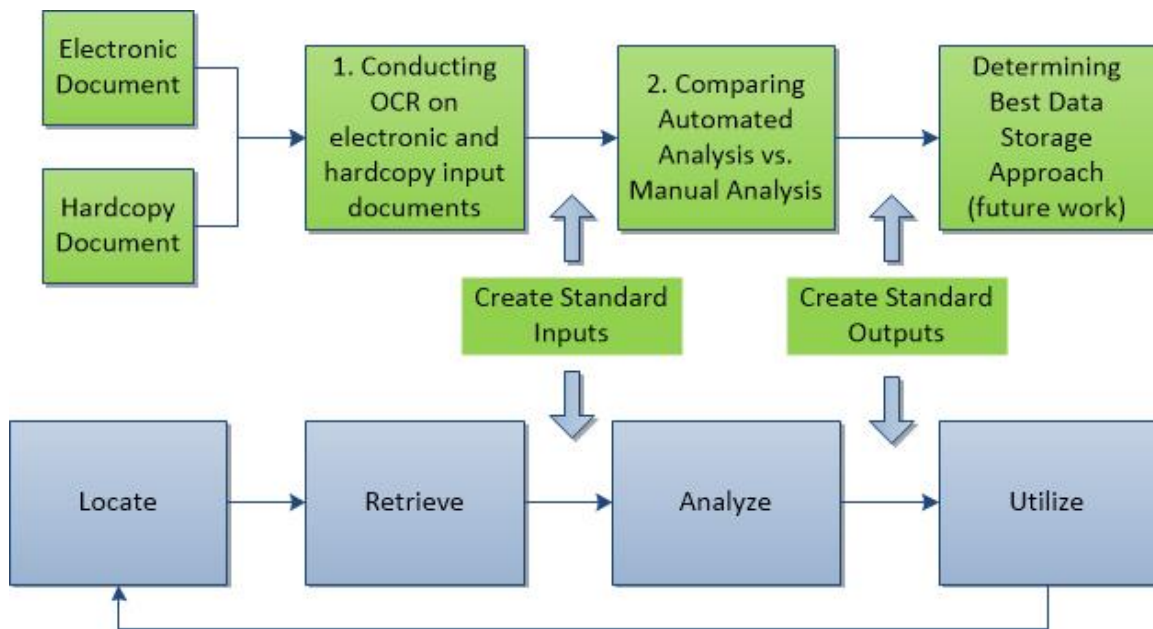


Figure 6. Consumption Data Correlation Process (refined)

The first question addresses the problem of storing both electronic and hardcopy documents. With consumption data appearing in both electronic and hardcopy

documents, planners must spend considerable time collecting and transcribing this data into one central location. Not only is this time-consuming and labor-intensive, critical consumption data may be omitted or incorrectly-interpreted. While conducting OCR can be labor-intensive at first glance, the use of an OCR application can reduce the administrative burden placed on the planner by (a) allowing them to convert hardcopy documents into electronic documents which can be searched and (b) allowing them to convert pre-existing electronic documents that can not be searched into searchable documents. These searchable documents can also be used as inputs to data extraction programs that have the ability to extract consumption data elements and provide them in a useful output. For example, a (key, value) pair for a database or a simple text file containing solely consumption data elements. In order to provide an accurate snapshot of OCR, three off-the-shelf OCR applications were compared: an open-source, freely-available, online application; a licensed version of Microsoft OneNote®; and a licensed version of Nuance OmniPage®. The accuracy rate, capabilities, and limitations for each application were tested and recorded in Chapter IV. As a byproduct of this comparison, a standard text file was created that was later used by the analysis programs. This text file was reviewed and corrected until its contents were 100% accurate, creating “perfect inputs” for the analysis programs.

To answer the second question, two simple analysis programs were created and compared. These programs used the text file outputs created by the OCR applications as inputs. The goal of the first program was to maximize the work done by the program and reduce the amount of user interaction necessary. While the program analyzes the input files using pre-defined logic statements, there is still a requirement for the end-user to review and verify the entries at the end. The goal of the second program was to compare an alternative approach, requiring continual user interaction throughout the decision-making and review process. This program was created in two versions: line-by-line and page-by-page. The byproduct of this comparison was the creation of a standard text file and a (key, value) pair in the form of a Python list data structure. These outputs represent the last step of the study. Determining the best data storage and presentation approach for these outputs is left for future research.

C. PRO/CON ANALYSIS OF AUTOMATION

On one hand, digitizing and storing consumption data has the following benefits:

- Reduction in manpower hours necessary to correlate data.
- Allows access to consumption data from anywhere in the world.
- The ability to provide consumption data in one location.
- Eliminates the need to keep documents in hardcopy form, providing cost savings and reducing environmental usage.
- The ability to present the consumption data in various electronic formats—database, charts, tables, text documents, etc.

On the other hand, there are drawbacks:

- Higher cost in database / application administration (personnel and equipment incurred).
- Requires secondary investment in security and network monitoring personnel and systems.
- Depending on the extensiveness of the system, it may require dedicated personnel to operate and maintain the system.
- Relying solely on the software may, as in the case of using historical data, erode the core competency of the end user.
- Data corruption and/or data loss could result in a lack of availability for the system or data.
- Should this process be completely digitized and the printing of hardcopy consumption documents be ceased, an end-user without access to the system would be unable to accomplish their tasks.

While these benefits and drawbacks may not be all encompassing, the progression towards a computer-aided system is a natural progression and is detailed in the following chapters.

D. ORGANIZATION OF THESIS

Chapter II conducted a background study by presenting two forms of OCR, free-form and template-based, and discusses several options for storing the output from the analysis programs. Chapter III discusses two approaches for analyzing electronic consumption documents: automated and walkthrough. Chapter IV compares the OCR applications mentioned in Chapter II and demonstrates the analysis programs presented in

Chapter III. Chapter V covers summary results of Chapter IV, lists recommendations reached as a result of this research, and illustrates areas for continued research.

II. BACKGROUND STUDY

A. WHAT IS OPTICAL CHARACTER RECOGNITION?

This field of study was summarized in 1993 by Line Eikvil, a Norwegian scientist who specialized in pattern recognition, computer vision, and text mining as follows:

Optical Character Recognition deals with the problem of recognizing optically processed characters. Optical recognition is performed off-line after the writing or printing has been completed as opposed to on-line recognition where the computer recognizes the characters as they are drawn. Both hand printed and printed characters may be recognized but the performance is directly dependent upon the quality of the input documents. [6]

Although there has been refinement and improvement in OCR technology, this summary still represents the fundamental principles behind the process. We are primarily concerned with a) performing optical recognition and b) ensuring high quality input documents are supplied to the process. The latter of which is a universally-expected norm—in order for any application to provide the best outputs, it must be given the best inputs. As a means to an end, consumption data must exist on a computer and be recognized by the analysis program. In order to do this for consumption documents, OCR was leveraged to handle two cases:

- **Consumption data contained in hardcopy documents.** Here, the document exists solely in hardcopy format and OCR must be conducted on the document to create an electronic version.
- **Consumption data contained in electronic documents but not recognized by applications as a machine-readable format.** Here, the document exists in an electronic version but the document (or parts of it) may be presented as data that an application can't interpret. For example, the Python programming language is unable to natively interpret Microsoft Word .docx extensions or data contained in PDF files.

1. OCR Software Approaches

In order to best understand the nature of OCR, we conducted OCR on a usage table and present the results as an example in Figure 7. Note: this data was originally presented in a vertical landscape view. For the purpose of the example, it was manually

converted to a horizontal profile view. Chapter IV addresses the ability and accuracy of OCR applications to conduct this procedure automatically.

Infantry-Heavy Threat Combat Planning Factors Table										
Weapon ID Sequence										
Weapon			Ammunition		GCE RATES			Other than GCE Rates		
Weapon ID	Nomenclature	DODIC	Nomenclature	Daily ASSAULT	Daily SUSTAIN	Basic Allowance	Daily ASSAULT	Daily SUSTAIN	Basic Allowance	
B0471	SQUAD DEMOLITION SET	M032	CHARGE, DEMO BLOCK 1 LB TNT	15.00753	3.39605	48	15.00753	3.39605	15	
B0471	SQUAD DEMOLITION SET	M130	CAP, BLASTING ELECTRIC	18.01945	4.00000	150	18.01945	2.03084	150	
B0471	SQUAD DEMOLITION SET	M131	CAP, BLASTING NON-ELECTRIC	60.00000	38.00000	260	60.00000	38.00000	180	
B0471	SQUAD DEMOLITION SET	M458	CORD, DETONATING PETN	1401.30498	340.45345	1500	1401.30498	340.45345	1500	
B0471	SQUAD DEMOLITION SET	M670	FUZE, BLASTING TIME	500.00000	218.00000	3000	380.00000	218.00000	1500	
B0471	SQUAD DEMOLITION SET	M757	CHARGE, ASSEMBLY DEMOLITION	15.00753	3.39605	50	15.00753	3.39605	50	
B0471	SQUAD DEMOLITION SET	M766	IGNITER, TIME FUSE BLASTING	90.00000	64.00000	390	60.00000	38.00000	180	
B0471	SQUAD DEMOLITION SET	ML03	FIRING DEV, DEMO MULTIPURPOSE	0.47673	0.30030	21	0.47673	0.30030	12	

Figure 7. Usage Table—Pre-OCR (from [2])

Using an OCR application, this table was processed and a Microsoft Word® document was created. Figure 8 represents this document with errors highlighted in yellow.

Weapon ID Sequence									
Weapon ID	Nomenclature	DODIC	Nomenclature	Daily ASSAULT	Daily SUSTAIN	Basic Allowance	Daily ASSAULT	Daily SUSTAIN	Basic Allowance
B0471	SQUAD DEMOLITION SET	M032	CHARGE, DEMO BLOCK 1 LB TNT	15.00753	3.39605	48	15.00753	3.39605	15
B0171	SQUAD DEMOLITION SET	M130	CAP, BLASTING ELECTRIC	18.01945	4.00000	150	18.01945	2.03084	150
B3471	SQUAD DEMOLITION SET	M131	CAP, BLASTING NON-ELECTRIC	60.00000	38.00000	260	60.00000	38.00000	150
B0171	SQUAD DEMOLITION SET	M458	CORD, DETONATING PETN	1401.30498	340.45345	1500	1401.30498	340.45345	1500
B30471	SQUAD DEMOLITION SET	M670	FUZE, BLASTING TIME	500.00000	218.00000	3000	380.00000	218.00000	1500
B0171	SQUAD DEMOLITION SET	M757	CHARGE, ASSEMBLY DEMOLITION	15.00753	3.39605	50	15.00753	3.39605	60
B0171	SQUAD DEMOLITION SET	M766	IGNITER, TIME FUSE BLASTING	90.00000	64.00000	390	60.00000	38.00000	180
B0471	SQUAD DEMOLITION SET	ML03	FIRING DEV, DEMO MULTIPURPOSE	0.47673	0.30030	21	0.47673	0.30030	12

Figure 8. Usage Table—Post-OCR (after [2])

In order to create the output in Figure 8, the OCR application went line-by-line through the input document, recognizing, interpreting, and transcribing characters as it went along. The output illustrates that the majority of the data was transcribed correctly. Most of the errors that occurred were related to the numerical values associated to the DODIC and the various rate values on the right-hand side. In particular, numerical values that contained decimals and were longer encountered the highest error rate. Of note, the OCR application was intelligent enough to place the output data into a table. This kind of intelligence helps to refine and preserve the data for later analysis. Although the

application generated quite a few errors, this may not be an issue for a user who is using OCR for a simple text conversion tool. However, this error rate is cause for concern in applications that must maintain accurate information. Before discussing the manner in which this error rate can be reduced, it is important to understand that two forms of OCR exist: free-form and template-based.

a. Free-form OCR

The output produced in Figure 8 represents the use of free-form OCR. This is the most common and default method for most OCR applications and the method used by this thesis. While it allows for maximum flexibility of input formats such as text, tables, images, and alphanumeric characters, it is “considered slow and inaccurate at times... however, using free-form will still significantly reduce the amount of errors due to miskeying during manual data entry” [7]. Although free-form was used in this thesis, should consumption data present itself in a predictable fashion in the future, another form of OCR may provide a higher degree of accuracy and throughput—template-based OCR.

b. Template-based OCR

Many institutions and corporations throughout the world use template-based OCR to conduct data entry for a variety of systems. One such data entry method has gained popularity in the last decade—mobile banking deposit. Many major banking institutions allow members to directly deposit paychecks to their accounts using mobile-banking applications. The only requirement is for the end user to have an end-device capable of running the application and creating or importing an image (photograph) of the check to be processed. Again, to best understand this method, an example is an appropriate venue. Figure 9 is a blank check for illustrative and discussion purposes.

NAME
ADDRESS
CITY, STATE ZIP

0123
01-23456789

DATE

PAY TO THE
ORDER OF

\$

DOLLARS

BANK NAME
ADDRESS
CITY, STATE ZIP

FOR

⑆012345678⑆ 01234567890123⑆ 0123

Bank Routing
Number

Bank Account
Number

Check
Number

Figure 9. Blank Check for Template-based OCR Discussion

In computing, a template is defined as “a computer document that has the basic format of something (a business letter, chart, graph, etc.) that can be used many different times” [8]. A check follows this definition, having features that are pre-defined and used many different times in almost all other checks. With regards to understanding template-based OCR of a check, a check has the following distinguishable features:

- Rectangular in shape, having four corners at 90-degree angles.
- Standard fields to denote data fields (e.g., DATE, PAY TO THE ORDER OF, \$, DOLLARS, and FOR).
- Magnetic ink character recognition (MICR) font information at the bottom of the check—e.g., Bank Routing Number, Bank Account Number, and Check Number. The MICR font is a standard of the American National Standards Institute (ANSI) and was specifically created for recognition on checks.

While financial institutions withhold their proprietary software procedures and capabilities, with an understanding of how template-based OCR operates, their processes can be demystified to provide an understanding of how this technology can be used to interpret consumption documents. First, the check to be processed must be filled out with all the necessary fields completed. Note: some companies have software capable of detecting when fields are not completed and provide error responses. Second, the check must be entered into the system by either taking a picture of it or scanning it using an input device. Some applications may direct the input of the check from within the

application (e.g., clicking deposit check prompts the user to take a picture of the front and the back of the check). Once the check has been placed in an electronic format, depending on the sophistication of the application, it may be processed at the end-device or sent to a data processing system to verify the accuracy of the check. This is the phase where template-based OCR processing occurs. During this processing, the following questions are answered:

- **Is the document a check?** This can be done by verifying the shape and length of the document as well as verifying the 90-degree angles are present. Some companies may even have the ability to detect when the corner of a check is torn, however, such information is simply not known due to the proprietary nature of the software. If the document is not a check, an error should be returned.
- **Are all the fields filled out?** This is done by observing the marks that occur within the standard data fields. For example, a valid date should be after the DATE label and a valid numerical value should be after the dollar sign (\$). If “\$ ILUVCOOKIES” was written on the check, the software should be intelligent enough to understand that ILUVCOOKIES is not a valid numerical value and return an error.
- **Is the accounting information at the bottom valid?** This is done by interpreting the numerical values in MICR font at the bottom. Comparison of the name on the check to the account holder information would be a likely check for authenticity. If the accounting information is not correct, an error should be returned.

Finally, based on the results of the processing, the user should be given an acceptance or rejection status of the overall transaction. Figure 10 illustrates this process from beginning to end.

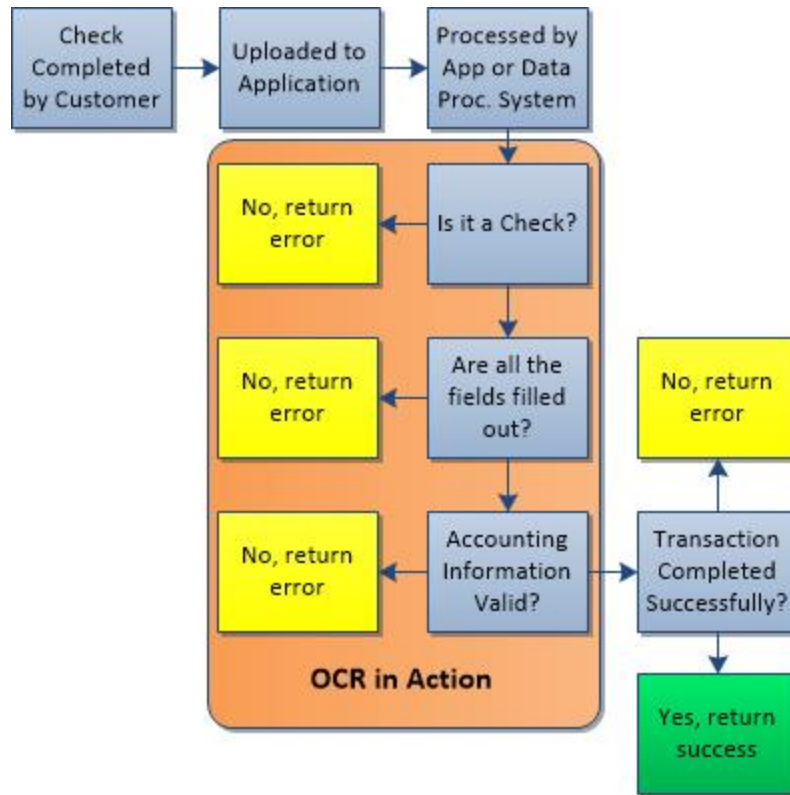


Figure 10. Template-based OCR Process of a Check

Throughout the process, the accuracy of the input (the check) is verified against a template (features of a check). Should the check deviate from the accepted template of a check or contain erroneous attributes, the software should default with an error. The important takeaway in this scenario is the need for template compliance, accuracy, and readability. In order for template-based OCR to interpret consumption data, the data should present itself repetitively and in the form of a template such as a table. Based on the consumption documents reviewed by this thesis, consumption data is a conglomerate of free text, tables, and lists, advocating the use of free-form OCR.

2. Increasing OCR Accuracy

Having discussed the various forms of OCR, we are able to return to the discussion of improving input accuracy with an appreciation for its importance. The usage of either free-form or template-based OCR is based on the input supplied to the OCR application. The effectiveness and accuracy of the application, independent of the

OCR format chosen, is based upon the clarity and readability of the given input. In computer science, there is often use of the acronym GIGO—garbage in, garbage out. Should the application for analyzing and interpreting consumption data receive bad inputs, it will most likely produce bad outputs. Thus, prior to converting consumption data using OCR, the highest emphasis must be placed upon ensuring the “cleanliness” of the input documents. In order to do so, the following steps should be taken:

(1) **Paper-to-electronic conversion.** Should a consumption document exist solely in hardcopy form, the best known copy should be used for OCR conversion. This document should have minimal interference and degradation. For example, if pages are torn, they should be replaced. Smudges, ink blots, and extra markings inside the document should be removed. OCR may attempt to recognize these markings and produce erroneous results. Note: photocopied documents tend to lose quality through blurring and fuzzing and should be used as a last resort.

(2) **Electronic-to-electronic conversion.** Should a consumption document exist solely in electronic form, the best known copy should be used as the input document. Again, the document should be inspected with the same regard as the paper to electronic conversion. Should the electronic document be of poor quality, another document should be used or the current one rewritten.

(3) **Document formatting.** For optimal processing, consumption data should be displayed in the appropriate format—text, table, columns, rows, etc. While OCR will attempt to identify and render this data intelligently, giving it the desired input for an expected result is recommended. For example, if you wanted the OCR application to create a table of data, provide it with a table of data. When possible, all pages should be presented in the same page layout. For example, all pages in the document should be presented in either portrait or landscape format—not a mixture of the two.

Although some of these steps may be manpower intensive up front, the dividends they pay in the long run may outweigh the costs incurred with auditing the OCR output. Reformatting and retyping a document may take several hours or even days. Likewise, the same amount of time may be spent auditing and correcting the OCR output if the OCR software misinterprets inputs and provides unreadable and/or gibberish outputs. For

example, the software may present 100 lines of data in portrait view when the original input was 3 lines of data presented in landscape view. Thus, it is incumbent upon the person who collects the electronic and hardcopy consumption documents for OCR to make the decision as to whether or not the document is in an acceptable state.

3. OCR Summary

OCR comes in two forms: free-form and template-based. Due to the nature of the consumption documents reviewed by this thesis, free-form OCR is leveraged. Emphasis is placed upon refining the input documents prior to OCR. In order to leverage OCR to its maximum effectiveness, it should be presented with the best inputs. The same mentality can be applied to the process of making wine: poor-quality grapes can seldom be mitigated by the winemaker.

B. METHODS FOR STORING CONSUMPTION DATA

The byproduct of the analysis programs compared in Chapter IV created two outputs: a standard text file and a (key, value) pair. To illustrate the rationale behind this decision, the following data storage methods are discussed: traditional databases and a document repository.

1. Traditional Database

A database can be leveraged to store consumption data. Two types of databases currently exist: relational and non-relational.

a. Relational Database Model

Inside of a relational database, data is represented in a schema (a framework), consisting of tables that have interconnecting relationships. Data may be spread across as few as one table or as many as thousands in order to correctly represent entities and relationships between the data elements. Each element (entity) in a database table has a unique identifier, referred to as the primary key, which prevents duplicate information from existing. Multiple entities are mapped together using relationship tables. In order to make this data available to the end-user, a database server is created and hosted, allowing

users to query the database. In order to query the database server, a user will typically build queries using structured query language (SQL) constructs. Figure 11 illustrates how one consumption data element may be represented in a relational database. Figure 12 is an illustration of the query submitted and the query's result based on the data shown in Figure 11.

Weapon Table (Entity)

<u>Weapon ID</u>	Nomenclature
<u>B0471</u>	Squad Demolition Set

Ammunition Table (Entity)

<u>DODIC</u>	Nomenclature
<u>M032</u>	Charge, Demo Block 1 LB TNT

GCE Rate Table (Entity)

<u>DODIC</u>	Daily ASSAULT	Daily SUSTAIN	Basic Allowance
<u>M032</u>	15.00753	3.39605	48

Other than GCE Rates Table (Entity)

<u>DODIC</u>	Daily ASSAULT	Daily SUSTAIN	Basic Allowance
<u>M032</u>	15.00753	3.39605	15

Part of Table (Relationship mapping between Weapon and Ammunition)

<u>Weapon ID</u>	<u>DODIC</u>
<u>M032</u>	<u>M032</u>

Rates for Table (Relationship mapping between Ammunition and Rates)

<u>DODIC</u>	Daily ASSAULT	Daily SUSTAIN	Basic Allowance	Daily ASSAULT	Daily SUSTAIN	Basic Allowance
<u>M032</u>	15.00753	3.39605	48	15.00753	3.39605	15

Figure 11. Consumption Data Element in Relational Form

select *

from Weapon, Ammunition, GCE Rate, Other than GCE Rates, Part of, Rates for

where Weapon.WeaponID = "B0471" and Ammunition.DODIC = "M032"

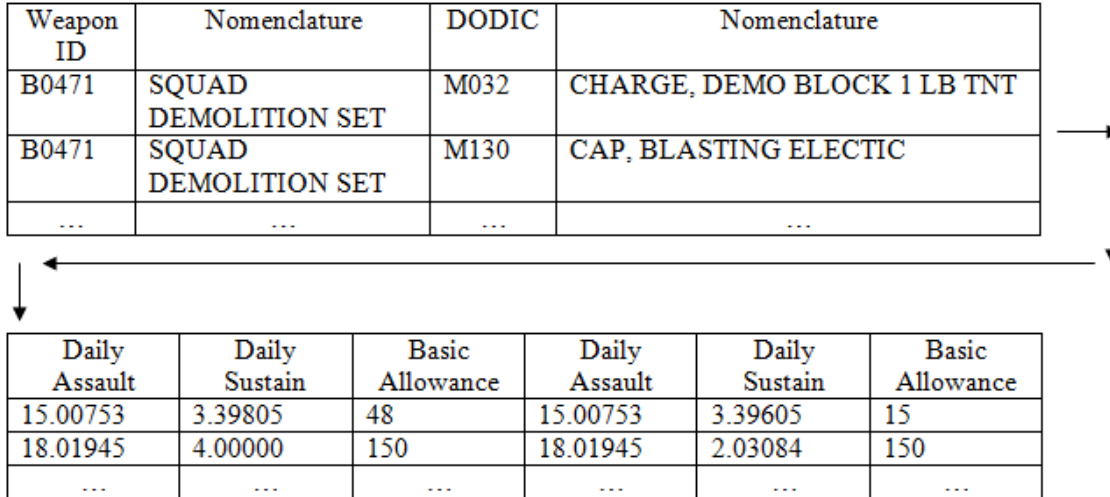


Figure 12. Relational Database Query and Result

Figures 11 illustrates the main drawback of implementing a relational database. Although only one consumption data element was given, six tables had to be created in order to correctly represent all of its data elements. While this may seem insignificant at first glance, this problem becomes more pronounced when data is spread across hundreds or thousands of tables. Here, a cost in computing time is incurred to search through each table, establish relationships, and present subsequent tables. Additionally, any inputs into the system must strictly adhere to the existing format, or "schema." Any inputs that deviate from the appropriate input format will be rejected or cause an error.

b. Non-relational Database Model

The main goal of using NoSQL ("Not Only SQL") is to break away from the problems associated with maintaining relationships in relational databases. Since a relational database must keep track of the relationships contained within the database, it must create extra tables in order to do so. This problem is avoided with NoSQL implementations by offering a variety of opportunities to store data. One of the common approaches is the use of a (key, value) pair to represent a single unique data element. This

(key, value) approach illustrates the use of a fundamental computer science data structure—the dictionary. The National Institute of Standards and Technology (NIST) define a dictionary as “an abstract data type storing items, or values. A value is accessed by an associated key. Basic operations for manipulating a dictionary are new, insert, find and delete” [5]. Figure 13 illustrates how one consumption data element may be represented in dictionary format. Figure 14 is an illustration of the query submitted in Java and the result which would be shown using the data shown in Figure 13.

Weapon Dictionary

Weapon ID (key)	Consumption Information (value)
B0471	SQUAD DEMO SET, M032, CHARGE, DEMO BLOCK LB TNT, 15.00753, 3.39605, 48, 15.00753, 3.39605, 15, M130, CAP, BLASTING ELECTRIC, 18.01945, 4.00000, 150, 18.01945, 2.03084, 150, ...

Figure 13. NoSQL (key, value) Dictionary Example

Program Code

```
public static void main(String[] args) {  
  
    HashMap map = new HashMap();  
  
    // This is where values are entered into the dictionary  
  
    map.put("B0471", SQUAD DEMO SET, M032, CHARGE, DEMO BLOCK 1  
    LB TNT, 15.00753, 3.39605, 48, 15.00753, 3.39605, 15, M130, CAP,  
    BLASTING ELECTRIC, 18.01945, 4.00000, 150, 18.01945, 2.03084, 150, ...);  
  
    String queryResult = map.get("B0471"); ← This is where values are retrieved  
  
    ... parsing and string manipulation code not shown for simplicity. In a real-world  
    application, parsing logic would require CPU resources of the end device  
    requesting information from the server.
```

Program Output

//While one form of potential output shown, there are unlimited possibilities of
representing this data to the end user.

Search results for "B0471" are:

WeaponID: B0471

Nomenclature: SQUAD DEMO SET

Ammunition:

Subcomponent 1:

DODIC: M032

Nomenclature: CHARGE, DEMO BLOCK 1 LB TNT

GCE Daily Assault: 15.00753

GCE Daily Sustain: 3.39605

GCE Basic Allowance: 48

Other than GCE Daily Assault: 15.00753

Other than GCE Daily Sustain: 3.39605

Other than GCE Daily Basic Allowance: 15

Subcomponent 2:

...

Subcomponent 3:

Figure 14. Java-implemented Dictionary and Query: Result

While a (key, value) implementation is discussed and used as the output format of the analysis program in this thesis, JavaScript Object Notation (JSON) and extensible markup language (XML) formats also exist.

c. Database Architecture

Figure 15 illustrates how a database can be implemented. Using the illustration as a guide, we further discuss each component of the database infrastructure in detail.

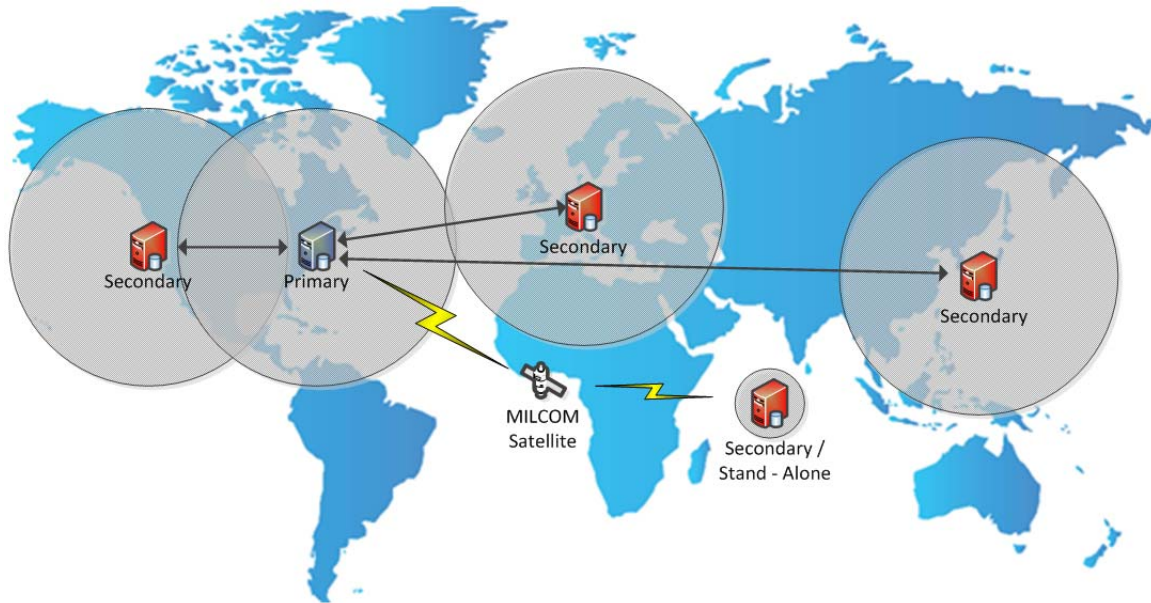


Figure 15. Global Database Architecture

The following components are necessary to implement a database:

- **Primary Server**–The primary server is responsible for providing each secondary server the most valid and updated information.
- **Secondary Server(s)**–To distribute the processing load off the primary server, secondary servers are created to handle transactions. Commonly, these servers are implemented in various geographic locations to not only distribute the load, but provide faster responses to the end user.
- **Secondary/Stand-Alone Server(s)**–Similar to the secondary server, a stand-alone server could be implemented at a forward operating base or onboard a ship in an amphibious ready group (ARG). Figure 15 illustrates the use of a server in an ARG environment. Here, updates can be sent back and forth between the stand-alone database and the primary server when connectivity is available over a military communications (MILCOM) satellite or other means. During periods of non-connectivity, elements in the database may become obsolete.
- **Intercommunication Capability/MILCOM Satellite**–Connections between the land-based primary and secondary servers could be done over a variety

of mediums—direct link, satellite, dedicated line, etc. Communications between a database at a FOB or an ARG requires reach-back capability via satellite communications using either a MILCOM satellite or commercial provider, such as INMARSAT. Such communication is essential to replicating changes from the primary database to each secondary implementation.

Database locations require additional hardware and software:

- Storage devices—required for storing the database software and data contained in the database itself.
- Network devices—provides connectivity in and out of the database.
- Input devices—mouse, keyboard, scanners, etc.
- Database software—necessary to run the database.

Figure 16 illustrates the necessary components for a database.

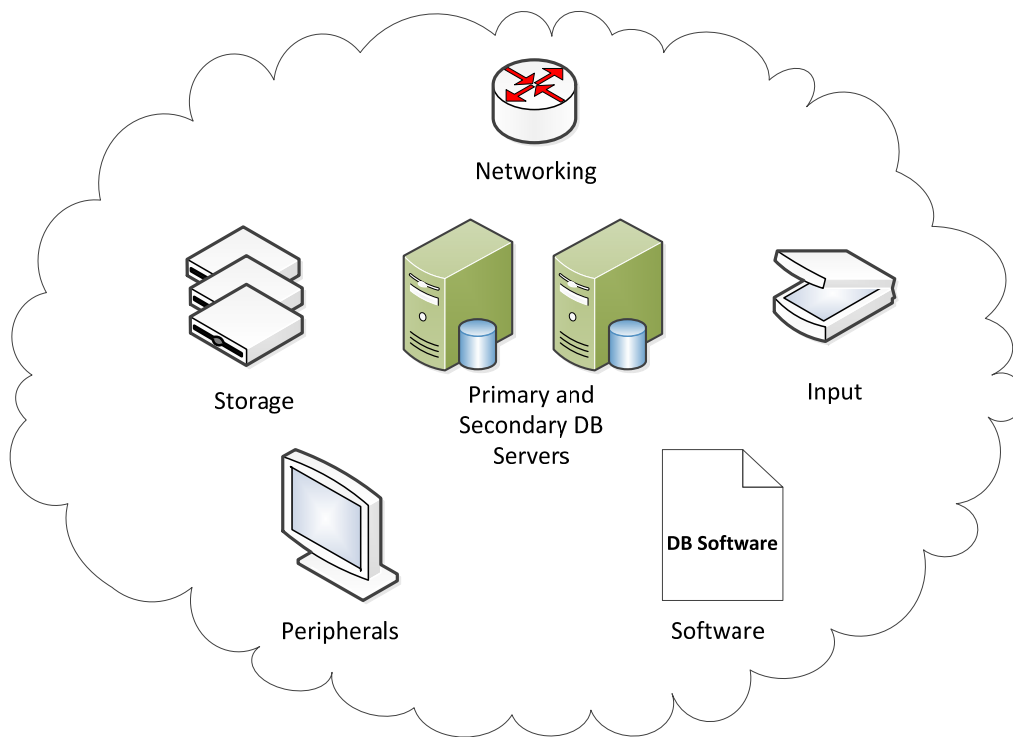


Figure 16. Database Location Hardware Instance

2. Document Repository

For the purpose of this discussion, we reference the Department of Defense (DOD) website that lists all valid DOD instructions. Figure 17 displays a portion of this site for illustrative and discussion purposes as shown in [4].

Copy to clipboard

CSV

Excel

PDF

Print

ISSUANCE NUMBER	ISSUANCE DATE	ISSUANCE SUBJECT	CHANGE #
DoDI 1000.01	4/16/2012	IDENTIFICATION (ID) CARDS REQUIRED BY THE GENEVA CONVENTIONS	CH 1
DoDI 1000.04	9/13/2012	FEDERAL VOTING ASSISTANCE PROGRAM (FVAP)	
DoDI 1000.11	1/16/2009	FINANCIAL INSTITUTIONS ON DOD INSTALLATIONS	
DoDI 1000.13	1/23/2014	IDENTIFICATION (ID) CARDS FOR MEMBERS OF THE UNIFORMED SERVICES, THEIR DEPENDENTS, AND OTHER ELIGIBLE INDIVIDUALS	
DoDI 1000.15	10/24/2008	PROCEDURES AND SUPPORT FOR NON- FEDERAL ENTITIES AUTHORIZED TO OPERATE ON DOD	

Showing 1 to 714 of 714 entries

Figure 17. DOD Instructions in Circulation (from [4])

This website gives the appearance of a database. The information is displayed in a table with columns. It has search and filter capability with full and partial-match functionality. The website is accessed via a compatible web browser such as Microsoft Internet Explorer, Mozilla Firefox, or Google Chrome. While this structure gives the appearance of a traditional relational or non-relational database, it has been created using JavaScript and HTML. When a hyperlink is clicked, the user is directed to the resource that is located in the web server's directory. Thus, no "query: result" is conducted. When the user clicks a hyperlink, an HTTP GET request is sent to the server. The web server then sends the material requested back to the user. In the case of this example, a GET

request returns a document that exists as a PDF in the server's directory. This method differs from a traditional database in the way data is stored. Unlike a database, an entire file is stored in a directory that can be accessed. Users are able to search and retrieve by document name but do not have the ability to search all the files at once.

3. Data Storage Summary

Traditional databases can be leveraged to store consumption data. Relational databases can maintain relationships between entities—"Weapon and Ammunition" for example, but may become cumbersome with large quantities of data. Non-relational databases circumvent this problem by relaxing input types and eliminating the need to maintain relationships between data elements. However, non-relational databases are less mature and few people are skilled to maintain them. Alternative methods for storing data exist. Storing consumption documents in text files on a web-server is one such alternative.

III. ANALYSIS PROGRAMS

A. PROGRAM CREATION

Once the consumption documents have been converted into an electronic format and collected in a centralized location by the planner, the process of analyzing the contents of these documents may begin. Currently, no programs exist that have the sole purpose of extracting consumption data elements. This is an important statement because it illustrates the infancy of such a program. Programs are often created by understanding and copying the core functionality and limitations of another similar program. Thus, we must look at this program from the ground up.

1. Language Selection

Each and every program in existence is created out of lines of code. These lines of code are written using a programming language. One such language, Python, is a widely-known and implemented language and will be used as the language of choice for this thesis. It can be explained easily (compared to the other languages) and avoids many restrictions that other languages must enforce.

2. Design and Functionality Specification

Figure 18 illustrates the flow of data in and out of the proposed extraction programs, providing a framework for discussion.

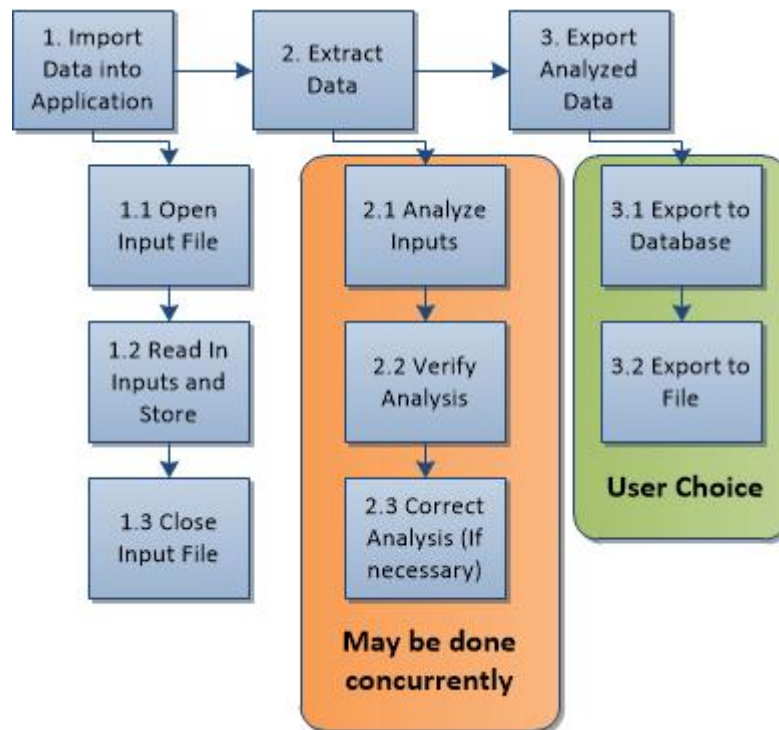


Figure 18. Application Flow

a. Data Import (1.1)—Open the Input File

The first step the program must take is the opening of an input file. A number of input file types may exist depending on the OCR software used to create them. However, these file types must be recognized by the program. To prevent any problems, the programs presented in this thesis used a simple text file, specifically the use of a .txt extension. Figure 19 illustrates a simple file-open routine that can be used by the program to open a file and count the number of lines it has. This is an important step because it ensures the program can conduct basic operations on the input file.

```

# Note: text following a pound sign (#) represents a
# comment and is not executable code.

# This program asks the end user to specify the name of an
# input document. Once the user has done so, the file is then
# opened and the number of lines are counted.

print ("Please enter the file name of the input document: ")

filename = input() # Here the user specifies the input name.

filename = filename + ".txt" # We append .txt file extension.

file = open(filename, 'r') # We open the file.

lineCount = 0

for line in file:
    lineCount = lineCount + 1 # And then count the number of lines.

print ("Line count for " + filename + " is: " + str(lineCount))

----- OUTPUT -----

Please enter the file name of the input document:

consumptiondata

Line count for consumptiondata.txt is: 3

```

Figure 19. Open Input File Routine with Line Counting Logic

This simple program first prompts the user for an input file name - “consumptiondata” in this example. Second, the program appends the .txt file extension. This can be removed if the user is restricted to entering a filename ending with .txt. Third, the file is opened in read-only mode specified by the “r” option. Other access options exist, namely the write option, which will be discussed in the “Export to File” section. Lastly, the program walks through the file and counts each line as it is encountered. This number is then printed to the screen. In this example, “consumptiondata.txt” contained three lines which the program correctly interpreted. Note that for simplicity, and in light of the OCR conversion to be discussed later, this rudimentary program restricts the user to files with a “.txt” extension; however, a more complex implementation might present the user with a “chooser” window (“modal box”) that allows the user to select (“choose”) a file from a drop-down list of files contained within a given directory (“folder”) stored within the host system.

b. Read in Inputs and Store, Close Input File (1.2 and 1.3)

Once the input file is open, data can be read out of the file and manipulated in the program’s memory space. Once the contents of the file have been read as input, the file can then be closed. Figure 20 illustrates a program that can handle these tasks.

```

# This program reads in the inputs from the open file
# and places them into a list for further processing.
# Once the contents have been read in, the input file is
# closed and contents of the list is printed.

consumptionData = []          # Python list data structure
listSize = 0                  # Counters for program termination
i = 0

for line in file:
    consumptionData.append(line) # Place each line in the list
    listSize = listSize + 1      # Keep track of the amount of elements

print ("Number of lines in the list is: " + str(listSize) + "\n")

file.close()                  # Close the input file, no longer needed

while (i < listSize):
    print (consumptionData[i])   # Print out each line of input from
    i = i + 1                   # the secondary data structure

----- OUTPUT -----

Number of elements in the list is: 3

B0471 SQUAD DEMOLITION SET    M032  CHARGE, DEMO BLOCK 1LB TNT    ...
B0471 SQUAD DEMOLITION SET    M130  CAP, BLASTING ELECTRIC      ...
B0471 SQUAD DEMOLITION SET    M131  CAP, BLASTING, NON-ELECTRIC  ...

```

Figure 20. Read-in and Storage of Inputs

This program begins by creating a Python list data structure named “consumptionData” with two program counters to control program execution and termination. Once these have been initialized, each line of the input file, “consumptiondata.txt,” is read into the list data structure. For tracking purposes, the size of the list is incremented as each new element is placed in the list. This allows us to keep

track of the size of the list in the variable “listSize.” After all the data elements have been read, the file is closed and all the elements are printed to the screen.

c. Analyze, Verify, and Correct Inputs (2.1–2.3)

At this point, the consumption data elements have been accessed by the program and can be further analyzed and manipulated without the need of the input file. Two different approaches may be used to handle this phase:

- **Walkthrough Analysis.** Using this approach, the end user is heavily-involved in the decision-making process on a line-by-line, paragraph-by-paragraph, or page-by-page basis. Having been read into the program-provided data structure, a line, paragraph, or page of potential consumption data is presented to the user. The user must then make a decision of “yes or no” that the item(s) presented is/are valid data element(s). If the user responds “yes,” the item(s) are transcribed into a secondary data structure—one which consists of all the valid data elements. If the user responds “no,” the element(s) are disregarded and the user is presented a new data set. Alternatively, the user could be presented an opportunity to access a given data element and manually enter the correct information, thereby ensuring the consumption data is consistent with the original source. The goal of this approach is to process the entire document without the need to go back through it repeatedly. Although this approach is time consuming, it aims to ensure all the data elements in the document are analyzed. Should the logic to automatically analyze the document not exist, this would be a feasible alternative.
- **Automated Analysis.** Under this approach, the end-user is involved in the decision-making process at the end after the program has made a “best effort” to autonomously extract all consumption data elements. This method is preferred over the walkthrough analysis only if the parsing logic is extensive and accurate. The end goal of using this method is to save time by allowing the program to quickly analyze the document and present a summary at the end. Once the summary has been populated, the end-user must then review the output for correctness. Should the parsing logic be subpar, the end-user may find it takes longer to correct the summary than it would be to conduct the walkthrough analysis.

Determining the best method to use—walkthrough or automated—depends on a wide variety of factors, both personnel and software-related. Answering these questions may help to address this conundrum:

- Is the OCR output file an accurate depiction of the original consumption document?
- Is the end user adequately trained to verify the analyzed data elements?
- Is the program mature enough to handle automated analysis?
- Is the parsing logic robust enough to recognize and extract all of the potential data elements?

Although both approaches are discussed, the automated program was specifically designed to handle consumption documents reviewed by this thesis. In order to create a more robust automated application, further document analysis and logic test creation must occur. However, the walkthrough analysis program could be used to analyze any input file since it treats a line of data unambiguously. Figure 21 shows the input values to the walkthrough analysis program. Figure 22 shows the program and its output.

```
This is not force consumption data.  
  
B0471 SQUAD DEMOLITION SET    M032 CHARGE, DEMO BLOCK 1LB TNT    ...  
  
These are random numbers: 1234567890.  
  
B0471 SQUAD DEMOLITION SET    M130 CAP, BLASTING ELECTRIC        ...  
  
?15456%$^@#$$@#^#@$!@#%@#5235213@#%@#
```

Figure 21. Walkthrough Analysis Inputs—Consumptiondata.txt

```

# This is a program to conduct walkthrough analysis.

initialConsumptionData = []      # Initial contents of the input file
finalConsumptionData = []        # Validated outputs
listSize = 0                     # Counters for program termination
finalListSize = 0
i = 0

# First, read in all the inputs into the initial list data structure.

for line in file:
    initialConsumptionData.append(line)
    listSize = listSize + 1

file.close()                     # Close the input file, no longer needed

# Afterwards, allow the end user to validate data elements

userResponse = ""                # A variable to store the user's response

while (True):
    print ("Is this a force consumption data element?")
    print (initialConsumptionData[i])
    userResponse = input()
    if (userResponse == "yes"):    # Only keep valid inputs
        finalConsumptionData.append(initialConsumptionData[i])
        finalListSize = finalListSize + 1
    i = i + 1
    if (i == listSize):           # Terminate the program
        break

i = 0

while (i < finalListSize):
    print (finalConsumptionData[i])    # Print out the valid outputs
    i = i + 1

----- OUTPUT -----

B0471 SQUAD DEMOLITION SET      M032 CHARGE, DEMO BLOCK 1LB TNT    ...
B0471 SQUAD DEMOLITION SET      M130 CAP, BLASTING ELECTRIC       ...

```

Figure 22. Program to Execute Walkthrough Analysis

First, the program reads the post-OCR inputs into a list data structure named “initialConsumptionData.” Once this is complete, the input file is closed. Afterwards, a loop control structure is used to cycle through all the data elements in the list. As each data element is presented, the end-user must make a “yes” or “no” decision that the element is a consumption data element. If the user enters “yes,” this data element is copied into the final data structure named “finalConsumptionData” and the process continues until no more elements exist. Elements that do not receive a “yes” decision are skipped and the process continues until no more elements remain to be considered. At the end, the final list of data elements, having been verified by the end user, is printed out.

In addition to the ability to individually validate each element, logic can be included to allow the user to correct each element as necessary, as noted above for the walkthrough analysis. For example, the question “is this element accurate?” can be prompted to the user, allowing the user to inspect each data element for accuracy. Should the element be inaccurate, the program would then allow the user to modify the value of the element prior to placing it in the output data structure. Although this presents a line-by-line review of the data elements, many different variations can be created. For example, instead of showing one element at a time to the end-user, a page of data can be presented with each line having an associated index number. The user could then indicate which lines are consumption data elements and they would be copied over in the same manner.

The second approach, an automated analysis program, utilizes the same consumption data example inputs shown in Figure 21. While the goal is to achieve automation, the program still requires the end-user to review the summary results created by the program prior to saving. Figure 23 illustrates how pre-defined logic statements can be used to facilitate automation for most of the program.

```

# This is a program to conduct automated analysis.

initialConsumptionData = []          # Initial contents of the input file
finalConsumptionData = []            # Validated outputs
listSize = 0                         # Counters for program termination
finalListSize = 0
i = 0

# First, read in all the inputs into the initial list data structure.

for line in file:
    initialConsumptionData.append(line)
    listSize = listSize + 1

file.close()                          # Close the input file, no longer needed

# Afterwards, allow the program to make a decision based on logic statements.

searchString = "SQUAD DEMOLITION SET" # Pre-defined force consumption element

for element in initialConsumptionData: # Inspect each element in the list
    if searchString in element:
        finalConsumptionData.append(element) # Found a match, add to output
        finalListSize = finalListSize + 1

while (i < finalListSize):
    print(finalConsumptionData[i])        # Print the outputs
    i = i + 1

    ----- OUTPUT -----

B0471 SQUAD DEMOLITION SET    M032 CHARGE, DEMO BLOCK 1LB TNT    ...
B0471 SQUAD DEMOLITION SET    M130 CAP, BLASTING ELECTRIC      ...

```

Figure 23. Program to Execute Automated Analysis

Similar to the walkthrough analysis program, this program begins by reading and storing inputs into a list data structure. One approach for designing logic statements is to create search strings. In this example, the string “SQUAD DEMOLITION SET” is used to represent a consumption data element from the Class V(W) planning factors table. After this has been defined, the program scans through each element of input and does a logical comparison between the input and the desired search string. If the program is

presented with a line which contains the pre-defined search string, it places that particular line into the output listing. This test can create false-positives and must be further defined. This simple example merely illustrates how the program can conduct its own logical comparison of inputs without the need for constant human interaction. It also helps to illustrate how one could approach logic creation. For example, one could create a consumption data dictionary containing an extensive amount of terms that can be searched and compared against. Once this dictionary is populated, the program could then test to see if any of those words existed in a particular line of consumption information. Words such as “table” and “factors” would be good candidates for the dictionary along with specific consumption data table names.

It is important to note that neither the walkthrough nor the automated analysis programs represent a panacea solution. Due to the infancy of such a program, it may be necessary to begin with more of a hands-on application such as the walkthrough analysis program, which evolves over time into more of an autonomous solution. As the pre-defined logic base of the automated program becomes more mature, the program will produce more reliable results. Refinement and standardization of the input documents will also help to increase reliability.

d. Export Analyzed Data (3.1 and 3.2)

Once the end-user has reached this part of the program, the user should be in a position where they have the refined consumption outputs. These outputs should be carefully-reviewed and corrected to be consistent with the original source document. Although we use the term “end-user” to relate to the user of the application, we wholeheartedly expect that this data has also gone through multiple levels of verification (i.e., “up the chain of command”). Once this information has been vetted, it can then be directed as input to a database or saved to a file.

(1) Export to Database

Under this approach, the program can leverage the use of pre-defined software packages that interact with database systems. For example, Python has a module package

named “_mysql” that allows it to interface with Oracle’s relational database software—MySQL.

(2) Export to File

Instead of sending the outputs to a database, they can be redirected to a file. In order to do this, the program simply opens a file in the write (“w”) mode, as previously mentioned. By saving to a file, many different options exist based on the output value format (e.g., .docx, .txt, .xml).

Having discussed methods for processing a file once created, the next chapter discusses options for OCR applications by which the files themselves may be generated from existing hardcopy or non-character-based files.

IV. FIELD DEMONSTRATION, TESTING, AND RESULTS

A. OCR TESTING AND COMPARISON

1. Testing Environment

Pages were selected from various consumption documents that illustrated the different types of data and layouts that are commonly encountered by the planner:

- Figure 24 illustrates the front page of a Marine Corps order. This page contains information that provides background information and usage procedures for the document. However, it also contains information that can be used to uniquely identify the document and can be extracted for use as the key in a (key, value) pair. This example will be referred to as EX-1.
- Figure 25 illustrates a page that contains a usage table. This table is presented in profile view but must be turned 90 degrees clockwise to a landscape view to read it. This example will be referred to as EX-2.
- Figure 26 illustrates a page that contains a usage table in portrait view. This table appears as an image in the input document. This example will be referred to as EX-3.
- Figure 27 illustrates a page that contains a usage table in portrait view. This table has been previously-created using table formatting from another text editor. This example will be referred to as EX-4.

Each application was graded on a scale of 1-3, one for poor, two for fair, and three for good, respectively. A summary of these scores is presented at the end of this section. The following factors were compared and graded for each application:

- **Accuracy Rate.** The accuracy rate is depicted as the number of errors per page. When a data element is not converted or converted incorrectly, we counted it as a single error. A data element is defined as one single entry or word. For example, “B0471 SQUAD DEMOLITION SET” illustrates four data elements: B0471, SQUAD, DEMOLITION, and SET. Grading of this criteria was based on three thresholds:
 - 0–50 errors per page. Awarded a three.
 - 50–100 errors per page. Awarded a two.
 - > 100 errors per page. Awarded a one.
- **Consistency.** Consistency was tested to see if the applications return regular results and whether or not they encounter the same errors when given the same input document. To test consistency, each page was

scanned with each OCR application 10 times. Grading of this criteria was based on the following:

- 8-10 OCR attempts produced the same/similar result. Awarded a three.
- 4-7 OCR attempts produced the same/similar result. Awarded a two.
- < 4 OCR attempts produced the same/similar result. Awarded a one.
- **Speed.** Speed was measured as the amount of pages that could be converted per hour. Speed also takes into account the time necessary for a user to make corrections. Thus, speed is also directly-related to the accuracy rate. Grading of this criteria was based on the following:
 - 45–60 pages processed per hour. Awarded a three.
 - 30–45 pages processed per hour. Awarded a two.
 - < 30 pages processed per hour. Awarded a one.
- **Ease of Use.** This relates to the user's ability to efficiently use the application. This factor is based on our use of the program and may not accurately represent a novice user. Time to master functionality of the program guided the following grading criteria:
 - < 5 minutes required to master the program. Awarded a three.
 - < 15 minutes required to master the program. Awarded a two.
 - > 15 minutes required to master the program. Awarded a one.
- **Functionality.** This relates to the application's ability to provide useful and helpful tools to the user—e.g. spellcheckers, input formats, output formats, etc. Grading of this criteria was based on the following:
 - The application had numerous input/output file types and included functionality that substantially increased end-user productivity. Awarded a three.
 - The application had several input/output file types and included functionality that increased end-user productivity. Awarded a two.
 - The application had limited input/output file types and included functionality that had a limited impact on end-user productivity. Awarded a one.



DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON, DC 20380-0001

MCO 8010.1E
C 392
15 Apr 97

MARINE CORPS ORDER 8010.1E

From: Commandant of the Marine Corps
To: Distribution List

Subj: CLASS V(W) PLANNING FACTORS FOR FLEET MARINE FORCE COMBAT
OPERATIONS

Ref: (a) Marine Corps Ground Ammunition War Materiel
Requirement (WMR) Determination (1995-1996) Study
Final Report (NOTAL)
(b) MCO P4400.39G (NOTAL)
(c) FMFM 4-1
(d) FM 9-6
(e) FM 9-13

Encl: (1) Explanation of the Scenario-Base Combat Planning
Factors Tables
(2) Infantry-Heavy Threat Combat Planning Factors Table
(3) Armor-Heavy Threat Combat Planning Factors Table
(4) Composite Combat Planning Factors Table
(5) Combat Planning Factors for Special Operations

(6) Artillery Ancillary Items

1. Purpose. To promulgate Class V(W) combat planning factors (CPF's) to support Fleet Marine Force (FMF) combat operations.

2. Cancellation. MCO 8010.1D.

3. Background. Reference (a) reports the results of the Marine Corps Class V(W) WMR Study (1995-1996). Reference (b) establishes Marine Corps policy governing requirements determination, acquisition, management, and distribution of war reserve materiel. References (c), (d) and (e) provide logistical doctrine and associated tactics, techniques, and procedures for Class V(W) support during combat operations.

4. Planning Factors. Factors to be used during initial planning for combat operations are explained in enclosure (1) and shown in enclosures (2) through (5).

a. CPF's reflect the anticipated expenditure of ground ammunition over designated time periods of combat operations. These rates represent the unconstrained requirement. Unconstrained requirements are based on approved force structure, weapon mix, anticipated duration of combat, and the anticipated intensity of conflict. Once Version 2.1 of the Ammunition Prepositioning and Planning System (APPS) is fielded, the APPS

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

Figure 24. EX-1, Marine Corps Order Front Page (from [2])

APR 15 1997

Infantry-Heavy Threat Combat Planning Factors Table
Weapon ID Sequence

Weapon ID	Weapon Nomenclature	DODIC	Ammunition Nomenclature	GCE RATES		Other than GCE Rates	
				Daily ASSAULT	Daily SUSTAIN	Daily ASSAULT	Daily SUSTAIN
B0471	SQUAD DEMOLITION SET	M032	CHARGE, DEMO BLOCK 1 LB TNT	15.00753	3.39605	48	15.00753
B0471	SQUAD DEMOLITION SET	M130	CAP, BLASTING ELECTRIC	18.01945	4.00000	150	18.01945
B0471	SQUAD DEMOLITION SET	M131	CAP, BLASTING NON-ELECTRIC	60.00000	36.00000	260	60.00000
B0471	SQUAD DEMOLITION SET	M458	CORD, DETONATING PETN	1401.30498	340.45345	1500	1401.30498
B0471	SQUAD DEMOLITION SET	M670	FUZE, BLASTING TIME	500.00000	216.00000	3000	360.00000
B0471	SQUAD DEMOLITION SET	M757	CHARGE, ASSEMBLY DEMOLITION	15.00753	3.39605	50	15.00753
B0471	SQUAD DEMOLITION SET	M766	IGNITER, TIME FUSE BLASTING	90.00000	54.00000	360	60.00000
B0471	SQUAD DEMOLITION SET	ML03	FIRING DEV, DEMO MULTIPURPOSE	0.47673	0.30030	21	0.47673
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M032	CHARGE, DEMO BLOCK 1 LB TNT	20.00000	5.00000	20	2.66024
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M131	CAP, BLASTING NON-ELECTRIC	211.55665	48.93008	500	211.55665
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M456	CORD, DETONATING PETN	120.00000	30.00000	450	120.00000
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M670	FUZE, BLASTING TIME	2.11527	0.48808	4	2.11527
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M757	CHARGE, ASSEMBLY DEMOLITION	30.00000	7.00000	30	30.00000
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M766	IGNITER, TIME FUSE BLASTING	1.05755	0.48808	12	1.05755
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	ML03	FIRING DEV, DEMO MULTIPURPOSE	29.18648	20.44450	1000	0.18644
B0589	COMBAT MOBILITY VEHICLE	A131	CTG, 7.62MM 4 & 1 LINKED	15.84271	11.06704	864	5.00000
B0589	COMBAT MOBILITY VEHICLE	B542	CTG, 40MM HEDP LINKED FOR MK19	2.75651	1.81975	16	48.93008
B0589	COMBAT MOBILITY VEHICLE	G828	GRENADE, LAUNCHER SMOKE IR	0.70095	0.13273	2	0.70095
B1288	LINE CHG LAUNCH KIT TRLR MTD	J143	ROCKET MOTOR, 5 INCH	0.70085	0.13273	2	0.70085
B1288	LINE CHG LAUNCH KIT TRLR MTD	M913	CHARGE, DEMO LINEAR HE	2.36044	0.47000	3	0.13273
B1315	LINE CHG LAUNCHER F/ AAVP7A1	J143	ROCKET MOTOR, 5 INCH	2.36044	0.47000	3	0.13273
B1315	LINE CHG LAUNCHER F/ AAVP7A1	ML25	CHARGE, DEMO LINEAR HE LVT	3.19475	1.30320	24	0.70085
E0150	BRIDGE ARMORED LAUNCHER	G828	GRENADE, LAUNCHER SMOKE IR	0.21895	0.04515	3	0.70085
E0207	COMMAND LAUNCH UNIT, JAVELIN	JAVL	JAVELIN	8.67150	21.03178	100	0.13273
E0311	DESIGNATED MARKSMAN RIFLE	AA11	CTG, 7.62MM BALL MATCH	0.61270	0.04680	318	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D003	CHARGE, SPOTTING PROJECTILE	0.43155	0.16044	0.73	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D501	PROJ, 155MM APERS ADAM-L	0.95734	0.40028	1.48	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D502	PROJ, 155MM APERS ADAM-S	0.27768	0.11678	0.7	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D505	PROJ, 155MM ILLUM	0.05965	0.02288	0.1	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D510	PROJ, 155MM COPPERHEAD	1.00468	0.23352	0.6	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D514	PROJ, 155MM RAAM-S	0.15045	0.36158	0.28	0.13273
E0665	HOWITZER MED TOWED 155MM M198	D515	PROJ, 155MM RAAM-L				

Figure 25. EX-2, Usage Table in Profile View (from [2])

TABLE 2. COMBAT PLANNING FACTORS FOR SPECIAL OPERATIONS

DODIC	NOMENCLATURE	QUANTITY PER	
		MEU(SOC)	
A011	CTG, 12 GA 00 BUCK		480
A014	CTG, 12 GA 7.5 SHOT		480
A023	CTG, 12 GA SLUG		250
A024	CTG, 12 GA LOCKBUSTER		300
A136	CTG, 7.62MM MATCH		460
A260	CTG, 9MM JHP	18,000	
A363	CTG, 9MM BALL	18,000	
A475	CTG, .45 CAL BALL	6,000	
AX14	SHOTGUN PRIMER	1,000	
DWBS	DIVERSIONARY CHARGE MK 141 MOD 0		500
L302	SIG CTG, WHITE FLARE		350
L304	SIG CTG, GREEN FLARE		350
L328	SIG CTG, RED FLARE		350
LX11	SIGNAL ROCKET LAUNCHER KIT		50
M031	CHG, DEMO, BLOCK, TNT 1/2 LB		192
M032	CHG, DEMO, BLOCK, TNT 1 LB		96
M039	CHG, DEMO, CRATERING		20
M130	CAP, BLAST ELEC		288
M131	CAP, BLAST NON-ELEC		480
M456	CORD, DETONATING, REIN	2,000 FT	
M670	FUZE, BLASTING, TIME, EXPLOSIVE LOADED	800 FT	
M766	IGNITER, TIME BLASTING FUZE		300
M980	CHG, DEMO, EXPLOSIVE SHEET 38 FT PER ROLL	2 RO	
M981	CHG, DEMO, EXPLOSIVE SHEET 25 FT PER ROLL	2 RO	
M982	CHG, DEMO, EXPLOSIVE SHEET 19 FT PER ROLL	2 RO	
ML03	FIRING DEVICE, DEMO, MULTIPURPOSE M124		112
MM30	CHG, DEMO 20 GRAMS		200
MM41	CHG, DEMO, SHAPED, FLEX LIN 30 GR/FT, 6 FT LENGTH		12
MM42	CHG, DEMO, SHAPED, FLEX LIN 40 GR/FT, 6 FT LENGTH		32
MM43	CHG, DEMO, SHAPED, FLEX LIN 60 GR/FT, 6 FT LENGTH		12
MM44	CHG, DEMO, SHAPED, FLEX LIN 75 GR/FT, 6 FT LENGTH		28
MM45	CHG, DEMO, SHAPED, FLEX LIN 100 GR/FT, 6 FT LENGTH		12
MM46	CHG, DEMO, SHAPED, FLEX LIN 225 GR/FT, 6 FT LENGTH		12
MM47	CHG, DEMO, SHAPED, FLEX LIN 400 GR/FT, 6 FT LENGTH		12
MM48	CHG, DEMO, SHAPED, FLEX LIN 600 GR/FT, 6 FT LENGTH		12
MM56	DUAL LEAD NONEL PRIMADET, 175 MS DELAY, 100 FT LENGTH		75
MN14	FIRING DEVICE HAND HELD MK 54		5

Figure 26. EX-3, Usage Table as an Image (from [2])

ARTILLERY ANCILLARY ITEMS

MCO 8010.1E

APR 15 1997

PROJECTILE		ANCILLARY ITEM		
DODIC	Nomenclature	DODIC	Nomenclature	Multiplier
D501	PROJ 155MM, ADAM-L, M692	D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
		(N285, FZ, MTSQ, M577 may substitute)		
D502	PROJ 155MM, ADAM-S, M731	N523	PRIMER, PERCUSSION, M82	1.1
		D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
D505	PROJ 155MM, ILLUM, M485A2	(N285, FZ, MTSQ, M577 may substitute)		
		(N248, FZ, MT, M565 may substitute)		
		N523	PRIMER, PERCUSSION, M82	1.1
		D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
D510	PROJ 155MM, COPPERHEAD, M712	D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
		(N285, FZ, MTSQ, M577 may substitute)		
		N523	PRIMER, PERCUSSION, M82	1.1
		D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
D514	PROJ 155MM, RAAM-L	D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
		(N285, FZ, MTSQ, M577 may substitute)		
		N523	PRIMER, PERCUSSION, M82	1.1
D515	PROJ 155MM, RAAM-S	D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
		(N285, FZ, MTSQ, M577 may substitute)		
D528	PROJ 155MM, SMOKE, WP, M825	N523	PRIMER, PERCUSSION, M82	1.1
		D532	CHG PROP 155MM, RED BAG, M203	0.2
		D533	CHG PROP 155MM, RB/WB, M119A1/	0.2
		D541	CHG PROP 155MM, WHITE BAG, M4	0.7
		N289	FUZE, ET, M762	1.05
		(N285, FZ, MTSQ, M577 may substitute)		
		N523	PRIMER, PERCUSSION, M82	1.1

ENCLOSURE (6)

Figure 27. EX-4, Usage Table as a Table (from [2])

2. OCR Applications

The following off-the-shelf programs were compared:

- **<http://www.onlineocr.net/>.** Free, open-source, and online OCR application. The primary focus of this application is to conduct OCR. Available in guest and member modes:
 - **Guest Mode.** Accepts PDF and image (JPG, BMP, TIFF, and GIF) with a maximum file size of 5 MB. Output file types are MS Word® .docx, MS Excel® .xlsx and plain text .txt. Conversion is limited to 15 documents (single page) per hour.
 - **Member Mode.** Accepts the same inputs as guest mode. Output file types are PDF, MS Excel® .xls and .xlsx, MS Word® .doc and .docx, plain text .txt, and a RTF document .rtf. Maximum input file size is increased from 5 MB to 100 MB. New members have a 25-page credit. Once this limit has been reached, additional pages must be purchased. The price-per-page decreases when bulk amounts are purchased. For example, purchasing 50 pages has a cost of 10 cents per page (\$4.99) whereas purchasing 50,000 pages has a cost of 0.4 cent per page (\$199.95).
- **Microsoft OneNote®.** Commercial software published by the Microsoft Corporation. Sold stand-alone or as a part of the Microsoft Office Suite®. The primary focus of this application is to provide a workspace for the user to collect notes and organize documents. OCR is a feature within OneNote®. Accepts any input file on the Windows OS: images, PDF, MS Office® document extensions, text files, etc. Output file extensions are: .doc, .docx, .txt, and .pdf.
- **Nuance OmniPage®.** Commercial software published by the Nuance Corporation. The primary focus of this application is to conduct OCR. Accepts digital camera images, images (JPG, BMP, TIFF, GIF, PNG), and PDF. There are over 50 different output file extensions that fall into eight categories: HTML, MS Excel®, MS Word®, MS PowerPoint®, PDF, RTF, Unicode Text, and XML.

Table 1 is a summary of the input file types accepted by the applications. Likewise, Table 2 illustrates the output file types that they are capable of producing.

	onlineocr.net		Microsoft OneNote®	Nuance OmniPage®
	Guest	Member		
Images (.jpg, .gif, .tiff, .png)	✓	✓	✓	✓
PDF (.pdf)	✓	✓	✓	✓
Text (.txt)			✓	
MS Word® (.doc, .docx)			✓	
MS Excel® (.xls, .xlsx)			✓	
Digital Camera (Direct input)				✓
Totals	2	2	5	3

Table 1. OCR Input File Types

	onlineocr.net		Microsoft OneNote®	Nuance OmniPage®
	Guest	Member		
PDF (.pdf)		✓	✓	✓
Text (.txt)	✓	✓	✓	✓
MS Word® (.doc, .docx)	✓	✓	✓	✓
MS Excel® (.xls, .xlsx)	✓	✓	✓	✓
MS PowerPoint® (.ppt)				✓
Rich Text Format (.rtf)		✓		✓
HTML				✓
XML				✓
Unicode Text				✓
Totals	3	5	4	9

Table 2. OCR Output File Types

Since the primary focus of OCR is detecting and transcribing text from images, it is unremarkable that each of the applications accept image files as input. However, each of them accepting a PDF file is important since this is the format in which the consumption documents will most likely be available. Also, it is important to note that OneNote® is capable of accepting many additional input file types because it conducts a file-to-image conversion of all input documents. For example, when a Word® document is placed into OneNote®, it represents the document as an image in the note. OCR must then be conducted on the image to extract the text.

Although all of the applications are capable of creating Word® and Excel® outputs, the text file extension, .txt, provides the most flexibility for developing the conversion programs. Creating a program to accept Word® and Excel® files adds no extra functionality and often requires unnecessary libraries. Therefore, this thesis will create a text file with a .txt extension as the output format of the OCR applications for later use in the data extraction programs.

3. Online, Open-Source OCR

The first application we tested was the open-source application available at <http://www.onlineocr.net/>. In order to maximize the number of documents that could be tested, the guest mode was utilized. The member mode offered no further extension of capability that would have been beneficial to the discussion. Figure 28 illustrates the main page of the application as of 8 August 2014.



Figure 28. <http://www.onlineocr.net>: Main Page and Features (after [9])

Although the application claims that it can convert a file up to 5 MB, it is important to note that it will only convert one page. If the input is a multi-page PDF, it will only convert the first page. Therefore, it was presented with each page until all the pages had been converted. The application converted EX-1 with zero errors. This result is unremarkable because the page was previously prepared using text editing software and presents a very clear input document. Figure 29 illustrates the post-OCR results of EX-1.



DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON, DC 20380-0001

MCO 8010.1E
C 392
15 Apr 97

MARINE CORPS ORDER 8010.1E

From: Commandant of the Marine Corps
To: Distribution List
Subj: CLASS V(W) PLANNING FACTORS FOR FLEET MARINE FORCE COMBAT OPERATIONS
Ref: (a) Marine Corps Ground Ammunition War Materiel Requirement (WMR) Determination (1995-1996) Study Final Report (NOTAL)
(b) MCO P4400.39G (NOTAL)
(c) FMFM 4-1
(d) FM 9-6
(e) FM 9-13
Encl: (1) Explanation of the Scenario-Base Combat Planning Factors Tables
(2) Infantry-Heavy Threat Combat Planning Factors Table
(3) Armor-Heavy Threat Combat Planning Factors Table
(4) Composite Combat Planning Factors Table
(5) Combat Planning Factors for Special Operations
(6) Artillery Ancillary Items

1. Purpose. To promulgate Class V(W) combat planning factors (CPF's) to support Fleet Marine Force (FMF) combat operations.

2. Cancellation. MCO 8010.1D.

3. Background. Reference (a) reports the results of the Marine Corps Class V(W) WMR Study (1995-1996). Reference (b) establishes Marine Corps policy governing requirements determination, acquisition, management, and distribution of war reserve materiel. References (c), (d) and (e) provide logistical doctrine and associated tactics, techniques, and procedures for Class V(W) support during combat operations.

4. Planning Factors. Factors to be used during initial planning for combat operations are explained in enclosure (1) and shown in enclosures (2) through (5).

a. CPF's reflect the anticipated expenditure of ground ammunition over designated time periods of combat operations. These rates represent the unconstrained requirement. Unconstrained requirements are based on approved force structure, weapon mix, anticipated duration of combat, and the anticipated intensity of conflict. Once Version 2.1 of the Ammunition Prepositioning and Planning System (APPS) is fielded, the APPS

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

Figure 29. Post-OCR Results of EX-1 using onlineocr.net (after [2])

EX-2 was tested next. Figure 30 illustrates the post-OCR results with errors highlighted in red.

Infantry-Heavy Threat Combat Planning Factors Table									
Weapon ID Sequence									
Weapon ID	Weapon Nomenclature	DODIC	Ammunition Nomenclature	GCE RATES			Other than GCE Rates		
				Daily ASSAULT	Daily SUSTAIN	Basic Allowance	Daily ASSAULT	Daily SUSTAIN	Basic Allowance
B0471	SQUAD DEMOLITION SET	M032	CHARGE, DEMO BLOCK 1 LB TNT	15.00753	3.39605	48	15.00753	3.39605	15
B0471	SQUAD DEMOLITION SET	M130	CAP, BLASTING ELECTRIC	18.01945	4.00000	150	18.01945	2.03084	150
60471	SQUAD DEMOLITION SET	M131	CAP, BLASTING NON-ELECTRIC	60.00000	36.00000	260	60.00000	36.00000	180
60471	SQUAD DEMOLITION SET	M456	CORD, DETONATING PETN	1401.30498	340.45345	1500	1401.30498	340.45345	1500
B0471	SQUAD DEMOLITION SET	M670	FUSE, BLASTING TIME	500.00000	216.00000	3000	380.00000	216.00000	1500
80471	SQUAD DEMOLITION SET	M757	CHARGE, ASSEMBLY DEMOLITION	15.00753	3.39605	50	15.00753	3.39605	50
60471	SQUAD DEMOLITION SET	M766	IGNITER, TIME FUSE BLASTING	90.00000	54.00000	390	60.00000	36.00000	180
130471	SQUAD DEMOLITION SET	ML03	FIRING DEV, DEMO MULTIPURPOSE	0.47673	0.30030	21	0.47673	0.30030	12
60472	DEMOLITION EQUIPMENT INDIVIDUAL	M032	CHARGE, DEMO BLOCK 1 LB TNT				2.66024	0.18644	5
130472	DEMOLITION EQUIPMENT INDIVIDUAL	M131	CAP, BLASTING NON-ELECTRIC	20.00000	5.00000	20	20.00000	5.00000	60
80472	DEMOLITION EQUIPMENT INDIVIDUAL	M456	CORD, DETONATING PETN	211.55965	48.93008	500	211.55965	48.93008	500
80472	DEMOLITION EQUIPMENT INDIVIDUAL	M670	FUSE, BLASTING TIME	120.00000	30.00000	450	120.00000	30.00000	500
80472	DEMOLITION EQUIPMENT INDIVIDUAL	M757	CHARGE, ASSEMBLY DEMOLITION	2.11527	0.48808	4	2.11527	0.48808	4
80472	DEMOLITION EQUIPMENT INDIVIDUAL	M766	IGNITER, TIME FUSE BLASTING	30.00000	7.00000	30	30.00000	7.00000	50
80472	DEMOLITION EQUIPMENT INDIVIDUAL	ML03	FIRING DEV, DEMO MULTIPURPOSE	1.05755	0.48808	12	1.05755	0.48808	4
B0589	COMBAT MOBILITY VEHICLE	A131	CTG, 7.62MM 4 & 1 LINKED	29.18648	2044456	1000			
80589	COMBAT MOBILITY VEHICLE	8542	CTG, 40MM HE-DB LINKED FOR MK19	15.94271	11.08704	864			
130589	COMBAT MOBILITY VEHICLE	G826	GRENADE, LAUNCHER SMOKE IR	2.79951	1.81975	16			
61298	LINE CHG LAUNCH KIT TRLR MTD	J143	ROCKET MOTOR, 5 INCH	0.70085	0.13273	2	0.70085	0.13273	2
81298	LINE CHG LAUNCH KIT TRLR MTD	M913	CHARGE, DEMO LINEAR HE	0.70085	0.13273	2	0.70085	0.13273	2
81315	LINE CHG LAUNCHER F/ AAVP7A1	J143	ROCKET MOTOR, 5 INCH	2.36044	0.47000	3			
81315	LINE CHG LAUNCHER F/ AAVP7A1	ML25	CHARGE, DEMO LINEAR HE LVT	2.36044	0.47000	3			
E0150	BRIDGE ARMORED LAUNCHER	G826	GRENADE, LAUNCHER SMOKE IR	3.19475	1.30320	24			
E0207	COMMAND LAUNCH UNIT, JAVELIN	JAVL	JAVELIN	0.21895	0.04515	3			
E0311	DESIGNATED MARKSMAN RIFLE	AK-1	CTG, 7.62MM BALL MATCH	8.67150	21.03178	100			
E0665	HOWITZER MED TOWED 155MM M198	D003	CHARGE, SPOTTING PROJECTILE	0.61270	0.04680	3.16			
E0665	HOWITZER MED TOWED 155MM M198	D501	PROJ, 155MM APERS ADAM-L	0.43155	0.18044	0.73			
E0665	HOWITZER MED TOWED 155MM M198	D502	PROJ, 155MM APERS ADAM-S	0.95734	0.40029	1.46			
E0665	HOWITZER MED TOWED 155MM M198	D505	PROJ, 155MM ILLUM	0.27768	0.11676	0.7			
60665	HOWITZER MED TOWED 155MM M198	D510	PROJ, 155MM COPPERHEAD	0.05965	0.02268	0.1			
E0665	HOWITZER MED TOWED 155MM M198	D514	PROJ, 155MM RAAM-S	1.00466	0.23352	0.6			
E0665	HOWITZER MED TOWED 155MM M198	D515	PROJ, 155MM RAAM-L	0.15045	0.36158	0.28			

Figure 30. Post-OCR Results of EX-2 using onlineocr.net (after [2])

When OCR was conducted on EX-2, 28 errors were encountered:

- Five errors occurred in the conversion of text.
- 23 errors occurred in the conversion of numbers.
- Of the 23 errors that involved numbers, 16 of those errors occurred in the first column, “Weapon ID.”

The reason for such a high error rate in the “Weapon ID” column can be attributed to the applications inability to distinguish the difference between the letter “B” and the numbers “6” and “8.” This was most likely caused by the fact that the letter “B” has rounded corners and appears fuzzy in the input document. This is normal and an expected degradation of a document whose original publish date was 15 April 1997. Also, the enclosures may have been adopted from a document that was created before the source document came into existence. An important finding is that the online OCR application was intelligent enough to determine that the data, although presented in a vertical fashion, was best suited for representation as a table in a horizontal view. Thus,

the application took the page that was originally presented in a profile view and presented its output as a file in landscape view.

EX-3 was tested next. Figure 31 illustrates the post-OCR results with errors highlighted in red.

TABLE 2. COMBAT PLANNING FACTORS FOR SPECIAL OPERATIONS

	MEU (SOC)	ANTITY PER
DODIC NOMENCLATURE		
A011 CTG, 12 GA 00 BUCK	480	
A014 CTG, 12 GA 7.5 SHOT	480	
A023 CTG, 12 GA SLUG	250	
A024 CTG, 12 GA LOCKBUSTER	300	
A136 CTG, 7.62MM MATCH	460	
A260 CTG, 9MM JHP	18,000	
A363 CTG, 9MM BALL	18,000	
A475 CTG, .45 CAL BALL	6,000	
AX14 SHOTGUN PRIMER	1,000	
DWBS DIVERSIONARY CHARGE MK 141 MOD 0	500	
L302 SIG CTG, WHITE FLARE	350	
L304 SIG CTG, GREEN FLARE	350	
L328 SIG CTG, RED FLARE	350	
LX11 SIGNAL ROCKET LAUNCHER KIT	50	
M031 CHG, DEMO, BLOCK, TNT 1/2 LB	192	
M032 CHG, DEMO, BLOCK, TNT 1 LB	96	
M039 CHG, DEMO, CRATERING	20	
M130 CAP, BLAST ELEC	288	
M131 CAP, BLAST NON-ELEC	480	
M456 CORD, DETONATING, REIN	20	
M670 FUZE, BLASTING, TIME, EXPLOSIVE LOADED	800 FT	
M766 IGNITER, TIME BLASTING FUZE	300	
ML03 FIRING DEVICE, DEMO, MULTIPURPOSE M124	112	
MM30 CHG, DEMO 20 GRAMS	200	
MM41 CHG, DEMO, SHAPED, FLEX LIN 30 GR/FT, 6 FT LENGTH	12	
MM42 CHG, DEMO, SHAPED, FLEX LIN 40 GR/FT, 6 FT LENGTH	32	
MM43 CHG, DEMO, SHAPED, FLEX LIN 60 GR/FT, 6 FT LENGTH	12	
MM44 CHG, DEMO, SHAPED, FLEX LIN 75 GR/FT, 6 FT LENGTH	28	
MM45 CHG, DEMO, SHAPED, FLEX LIN 100 GR/FT, 6 FT LENGTH	12	
MM46 CHG, DEMO, SHAPED, FLEX LIN 225 GR/FT, 6 FT LENGTH	12	
MM47 CHG, DEMO, SHAPED, FLEX LIN 400 GR/FT, 6 FT LENGTH	12	
MM48 CHG, DEMO, SHAPED, FLEX LIN 600 GR/FT, 6 FT LENGTH	12	
MM56 DUAL LEAD NONEL PRIMADET, 175 MS DELAY, 100 FT LENGTH	75	
MN14 FIRING DEVICE HAND HELD MK 54		

Figure 31. Post-OCR Results of EX-3 using onlineocr.net (after [2])

When OCR was conducted on EX-3, 37 errors were encountered:

- 33 errors occurred because the application omitted three entire lines of data.

- Two errors occurred in the conversion of text.
- Two errors occurred in the conversion of numbers.

An important finding is that the OCR application did not place the data elements into a table. The application interpreted the page contents as an image, rather than table. However, an appropriate amount of space was placed between the data elements for readability. The cause for the omission of three lines of data is unknown.

EX-4 was tested next. Figure 32 illustrates post-OCR results with errors in red.

MCO 8010.1E
APR 15 1997

ARTILLERY ANCILLARY ITEMS

PROJECTILE		ANCILLARY ITEM		
DODIC	Nomenclature	DODIC	Nomenclature	Multiplex
D501	PROJ 155MM, ADAM-L, M692	D533	CHG PROP 155MM, RBIWE , M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
			(N285, FZ, MTSQ, M577 may substitute)	
		N523	PRIMER, PERCUSSION, M82	1.1
D502	PROJ 155MM, ADAM-S, M731	D533	CHG PROP 155MM, RB/ WE , M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
			(N285, FZ, MTSQ, M577 may substitute)	
		N523	PRIMER, PERCUSSION, M82	1.1
D505	PROJ 155MM, ILLUM, M485A2	D533	CHG PROP 155MM, RBIWE , M119A11	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
			(N285, FZ, MTSQ, M577 may substitute)	
			(N248, FZ, MT, M565 may substitute)	
		N523	PRIMER, PERCUSSION, M82	1.1
D510	PROJ 155MM, COPPERHEAD, M712	D533	CHG PROP 155MM, RBIWE , M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N523	PRIMER, PERCUSSION, M82	1.1
D514	PROJ 155MM, RAAM-L	D533	CHG PROP 155MM, RBIWE , M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
			(N285, FZ, MTSQ, M577 may substitute)	
		N523	PRIMER, PERCUSSION, M82	1.1
D515	PROJ 155MM, RAAM-S	D533	CHG PROP 155MM, RBIWE , M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762	1.05
			(N285, FZ, MTSQ, M577 may substitute)	
		N523	PRIMER, PERCUSSION, M82	1.1
D528	PROJ 155MM, SMOKE, WP, M825	D533	CHG PROP 155MM, RED BAG, M203	0.2
		D533	CHG PROP 155MM, RBIWE , M119A1/	0.2
		D541	CHG PROP 155MM, WHITE BAG, M4	0.7
		N289	FUZE, ET, M762	1.05
			(N285, FZ, MTSQ, M577 may substitute)	
		N523	PRIMER, PERCUSSION, M82	1.1

ENCLOSURE (6)

Figure 32. Post-OCR Results of EX-4 using onlineocr.net (after [2])

When OCR was conducted on EX-4, 12 errors were encountered:

- Five errors occurred in the conversion of text.
- Seven errors occurred in the conversion of special characters.

The application was unable to determine the difference between the letter “D” and the number “0” due to degradation of the source document. The application was unable to determine the difference between the forward slash character “/” and the letter “I.” An important finding is the fact that the application recreated the table structure from EX-4 near-perfectly.

Table 3 illustrates a summary of the accuracy rate and important findings for the open-source application.

Page	Total Errors	Word Errors	Number Errors	Cause / Findings
EX-1	0	0	0	<ul style="list-style-type: none"> • 100% accuracy rate of conversion may be attributed to the previous use of text editing software.
EX-2	28	5	23	<ul style="list-style-type: none"> • Problems distinguishing the letter “B” from the numbers “6” and “8.” • Text converted from portrait view to landscape. • Data placed in a table data structure.
EX-3	37	5	32	<ul style="list-style-type: none"> • Text interpreted as an image rather than a table.
EX-4	12	8	4	<ul style="list-style-type: none"> • Near-perfect table recreation • Problems distinguishing the letter “D” from the number “0.” • Problems distinguishing the special character, forward slash “/” from the letter “I.”
Totals	77	18	59	

Table 3. Online, Open-Source OCR Results and Findings

Overall, the online and open-source OCR application demonstrated the ability to convert the original consumption documents into useful output files. It also had the ability to recreate tables and recognize when data is best presented in other formats. For example, converting the data contained in EX-2 from a vertical profile view to a horizontal landscape view is helpful. Based on the findings for this application, the following scores were given:

- Accuracy: 2
- Consistency: 3
- Speed: 1
- Ease of Use: 3
- Functionality: 1

The accuracy rate of the application was manageable and consistent. The application suffers in speed, limiting the user to 15 pages per hour which would be mitigated by using the member mode albeit while impacting the cost. Using and understanding the functionality of the program can be accomplished in five minutes. The application provides the least amount of input and output formats and has no spellchecking ability. Once OCR has been completed, the user must open the output file in a text editor to review and correct its contents.

4. Microsoft OneNote® OCR

Microsoft OneNote® was tested next. Figure 33 illustrates the post-OCR results of EX-1.

DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON, DC 20380-0001

Figure 33. Post-OCR Results of EX-1 using OneNote® (after [2])

Although OneNote® was given the same page that the online and open-source was given, it was unable to OCR the document past the first three lines. This is

remarkable because it represents the first major failure of an OCR application to successfully convert an input document. To explore the significance of different file formats, the input document was converted from PDF to a JPG. Figure 34 illustrates the second round of OCR testing conducted on EX-1 in an image format.

DEPARTMENT OF THE NAVY
MtAJUU*(I i i l S U N I I C O 51*155 5*4155 CUera
W*SMINCICS, OC 2VSIU4ULT
MIO 60101E
C 092
15 Apr 97
NASILIE CORPS OEDtI 5010.15
Fron Coneendant of the Merme torpe
To: Distribution List
Subj CLASPV (W) PtA191 L NG FACTORS FOR FtÆEetMPR jEX FORCE CCF91AT
OPERA?! f915
Rat: la) Marina Carp. Uraund Mnunitlen War Material|
Reguiranent IW4RI Detemination (1995-1996) Study
Final Report IWTAt)
lb) MCO D440039c (NOtal
lr) SWIM 4-I
ld) 574 9-6
ls) SW 9-13
anti: li) Nxplanatinn at iba Sranario-5.aaa Cat Planning
Factors Tables
2) Infantry-Heavy Threat Combat Planning Pactare table
II) Arnor-Heavy Thraat Cneat Planning Factors Tabla
li) Canpoatte Ccnbst Planning Factors taSte
IS) Ccnbst Planning Factors (or Special Operat ions
161 Artillery Ancillary ttens

Figure 34. Post-OCR Results of EX-1 using OneNote®: Second Pass (after [2])

Without listing the entire contents of the output, we can quickly see that the output is highly inaccurate both in spelling and format. Thus, in order to make an accurate output file, the post-OCR results would need to be heavily corrected. OneNote® has the built-in functionality of a spellchecker. This can be leveraged to correct the errors and alleviate some of the burden on the user, however, the user must select this option since it does not turn on automatically after OCR is complete.

EX-2 was tested next. Figure 35 illustrates the post-OCR results.

hICO1O.1E
 IPI
 Huh Hill
 0.13
 e- ll
 oO
 'I Z
 lilli11111 u
 øgcog— ce ce . . ; ceodo
 UIII! hHī h !
 00 0 O e 0 000 .0
 -
 *
 ilihil',
 I ' ;cjl 1
 '1
 0L e.e
 l
 e- e- e-
 l mll
 Hl
 J HšH ..
 44
 .-—
 H
 gg x
 !!!
 „ ..
 .c •e
 l
 .-..—
 3
 I i j !llifl
 ENCLOSLE (2

Figure 35. Post-OCR Results of EX-2 using OneNote® (after [2])

When OCR was conducted on EX-2, OneNote® was unable to accurately process the input. Although the document was converted to an image and placed back in OneNote® in a landscape view as a secondary test, the same result was encountered. Thus, heavy modification or reformatting of the input document would be necessary to properly process the document.

EX-3 was tested next. Figure 36 illustrates the post-OCR results.

MCO 8010.1E
 2. COMBAT PLANNING FACTORS FOR SPECIAL OPERATIONS
 QUANTITY PER
 DOD1C NOMENCLATURE MEU t SOC)
 P.011 CT'G, 12 GA 00 BUCK 460
 P.014 CTG. 12 GA 7.5 SHOT 480
 P.023 CTG, 12 GA SLUG 250
 P.024 Cto, 12 GA LOCKB&TSTER 300
 P.136 CTC, 7.62MM MATCH 460
 P.260 CTG, 9H14 3M7 18, 000
 P.363 CT'J, 9MM BALL 18,000
 P.475 CTG. .45 CAL. BALL 6,000
 AX14 SHOTGUN PRIMER 1,000
 DWBS DIVERSIONARY CHARGE 14K 141 MOD 0 500
 L302 SICCTG, WHITE FLARE 350
 L304 BIG CTG, GREEN FLARE 350
 L328 BIO CTG, RED FLARE 350
 LX11 SIGNAL ROCKET LAUNCHER KIT 50
 M03] CRO, DEMO, BLOCK. TNT 1/2 LB 192
 11032 CRO, DEMO, BLOCK. TNT 1 LB 96
 11039 CRO, DEMO, CRATERING 20
 11130 CAP. BLAST ELEC 288
 11131 CAP, BLAST NON-ELEC 480
 11456 CORD, DETONATING, REIN 2,000 PT
 11670 FUZE, BLASTING, TIME, EXPLOSIVE LOAD80 800 FT
 11766 IGNITER, TIME BLASTINGP UZE 300
 11980 CR6, DEMO, XXPOSIVESHEET 38 FT PER ROLL 2 RO
 11581 CRO, DEMO, EXPLOSIVESHEET 25 FT PER ROLL 2 RO
 11902 CHG, DEMO, EXPLOSIVE SHEET 19 FT PER ROLL 2 RO
 11103 FIRINGDEVICE, DEMO, MULTIPURPOSEM124 112
 1(1130 CRO, DEMO 20 CRAMS 200
 14114) CRO, DEMO, SHAPED, FLEX LIN 30 OR/PT, 6 PT LENGTH 12
 10142 CRO, DEMO, SHAPED, FLEX LIN 40 OR/PT, 6 FT LENGTH 32
 10143 CRO, DEMO, SHAPED. FLEX LIN 60 GR/FT, 6 FT LENGTH 12
 10144 CRO] DEMO, SHAPED, FLEX LIN 75 On/PT, 6 PT L(GTH 28
 141145 CRO, DEMO, SHAPED. FLEX LIN 100 GR/FT, 6 FT LENGTH 12
 141146 CRO, DEMO, SHAPED, FLEX LIN 225 GR/FT, 6 PT LENGTH 12
 141147 CHG, DEMO, SHAPED, FLEX LIN 400 GR/FT, 6 FT LENGTH 12
 11H46 CRO, DEMO, SHAPED, FLEX LIN 600 OR/PT, 6 FT LENGTH 12
 14H56 DUAL LEAD NONELP PRIMADET, 175 IS DELAY, 100 FT LENGTH 75
 FIN14 FIRING DEVICE HAND MELD 14K 54 S
 ENCLOSURE (5)
 2

Figure 36. Post-OCR Results of EX-3 using OneNote® (after [2])

When OCR was conducted on EX-3, 47 errors were encountered:

- 32 errors occurred in the conversion of numbers.
- 14 errors occurred in the conversion of text.
- One error occurred in the conversion of the special characters.

The majority of the errors occurred in the first column where the DODIC is represented by alphanumeric characters. The second region that encountered the most

problems also involved alphanumeric characters. An important finding is that OneNote® provided very little in the way of formatting. The original input had large areas of white space to provide readability whereas OneNote® left-aligned the majority of the document and removed this white space, making the output difficult to read for an end-user.

EX-4 was tested next. Figure 37 illustrates the post-OCR results.

```
ARTILLERY ANCILLARY ITEMS
MCC 8010.IE
APR 15
0502 PROJ 155MM. ADAM-S. M731
'A. NCILIARY ITEM
omed cloure
CHG PROP 15544M, RB/WB, M11BA1!
C140 PROP 155MM, GREEN BAG, M3
CHG PROP 155MM. WHITE BAG. M4
FUZE, ET. 14762
(N285, FZ. MTSO, 14577 may substitute)
PRIMER. PERCUSSION. 1452
C)IG PROP 15544M, RB/WB. M11QA1!
C140 PROP 15544M. GREEN BAG. M3
CHO PROP 15544M, WHITE BAG. 144
FUZE. EF. 14762
N285, FZ. MTSO. 14577 rn.y substklite)
PRIMER. PERCUSSION, 1482
CHG PROP 155MM. RBW8, M11BA1!
CHG PROP 155MM, GREEN BAG, M3
CHG PROP 155MM, WHITE BAG, M4
FUZE.. ET, 14762
(N285, FZ. MTSO, 14577 may substitute)
(N248. FZ. MT, M565 may substitute)
PRIMER. PERCUSSION, 1482
0533 01G PROP 155MM. RBW, MI 19A1!
0640 C1tOP PROP 15544M, GREEN BAG. M3
0541 04G PROP 15544M, WHITE BAG. 144
N523 PRIMER. PERCUSSION M62
D533 CHG PROP 15544M, RB(W8, 14119M!
0640 04G PROP 15544M. GREEN BAG. 143
0541 CHG PROP 15644M, WHITE BAG, 144
N289 FUZE. ET. 14762
```

Figure 37. Post-OCR Results of EX-4 using OneNote® (after [2])

When OCR was conducted on EX-4, two errors were encountered that involved the conversion of text. Although the input had been previously prepared using a text editor and was presented in a highly structured table format, OneNote® failed to recreate the table and present the data in a useful format. Table 4 illustrates a summary of the accuracy rate and important findings for OneNote®.

Page	Total Errors	Word Errors	Number Errors	Cause / Findings
EX-1	> 100	> 100	> 100	<ul style="list-style-type: none"> • Unable to convert the input document past the first three lines of data. • Native spellchecking capability discovered.
EX-2	> 100	> 100	> 100	<ul style="list-style-type: none"> • Complete failure to detect input layout
EX-3	47	14	32	<ul style="list-style-type: none"> • Words comprised of alphanumeric characters represented 46 out of 47 errors. • Problems distinguishing the special character, forward slash “/” from the special character exclamation point “!”.
EX-4	2	2	0	<ul style="list-style-type: none"> • Failure to recreate table for readability.
Totals	> 100	> 100	> 100	

Table 4. OneNote® Results and Findings

In summary, OneNote® was incapable of accurately conducting OCR on EX-1, EX-2, and EX-4. OneNote® comes with the functionality of spellcheck but does not provide the functionality to apply formatting to the OCR output. While the program offers unlimited OCR capability, it must be purchased in order to do so. The main benefit of the program remains focused on its ability to quickly and efficiently take notes and requires its secondary OCR feature to be further refined before wide-scale use as a reliable OCR application. Based on the findings for this application, the following scores were given:

- Accuracy: 1
- Consistency: 1
- Speed: 1
- Ease of Use: 2
- Functionality: 2

The accuracy rate of the application was poor and inconsistent. The application is capable of conducting OCR quickly; however, overall speed suffers based on the amount of errors that need to be corrected by the user. The built-in spellchecker functionality can

aid the user in correcting these errors and offsets some of the speed penalties. Using and understanding the functionality of the program can be accomplished in 10-15 minutes. The application provides the most input file types but has limited output options.

5. Nuance OmniPage® OCR

Nuance OmniPage® was the last application tested. When the software loads, the user is presented with a menu to choose what kind of conversion they would like to accomplish. Figure 38 illustrates this screen.

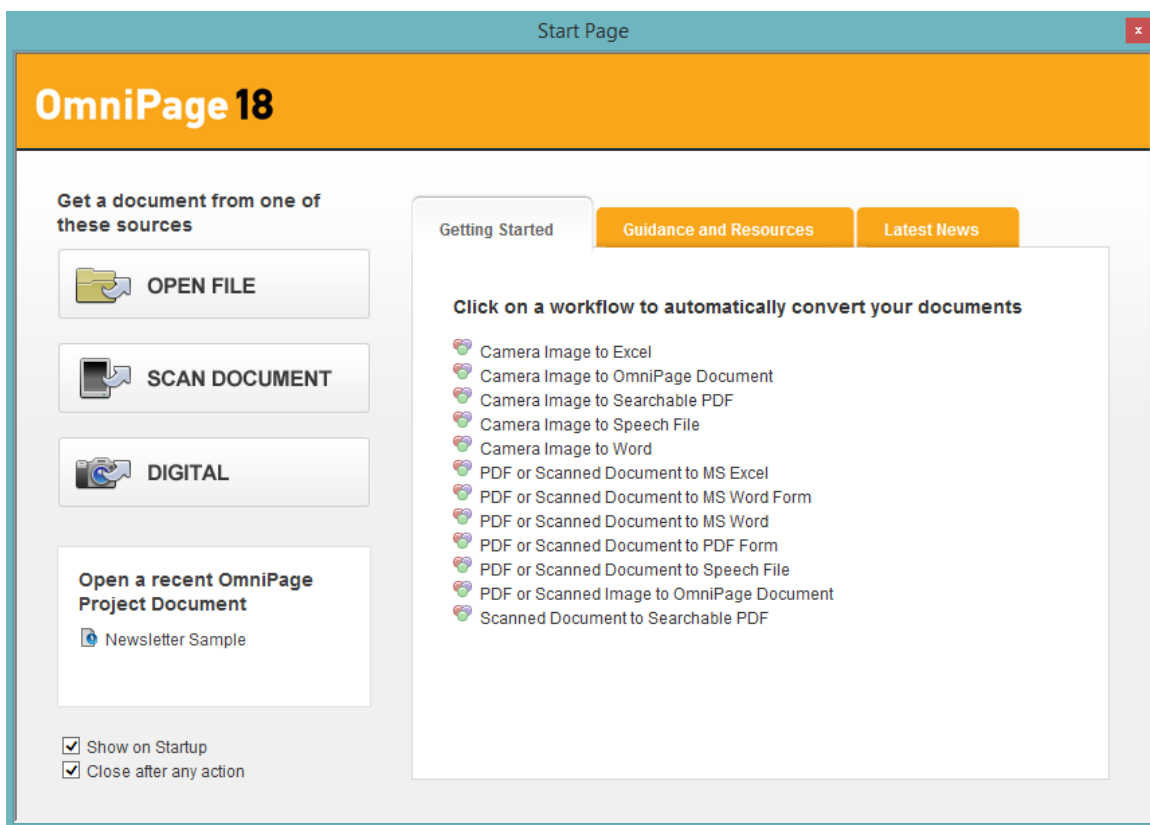


Figure 38. OmniPage Start Screen

While there are templates, known as “workflows,” the method used by this thesis was the “open file” option. Once this option is clicked, a dialog box is presented that allows the user to select what documents they would like to OCR. An important finding is that the application allows the user to select multiple input documents of various formats at one time. For example, you can select a PDF, an image, and another PDF all at

the same time. Likewise, you can select a PDF and it will import all pages of the PDF. After the files have been selected and imported into the program, the user is presented with a workspace view. The workspace view has multiple frames and allows the user to rearrange their frames as they please. This view is presented in Figure 39.

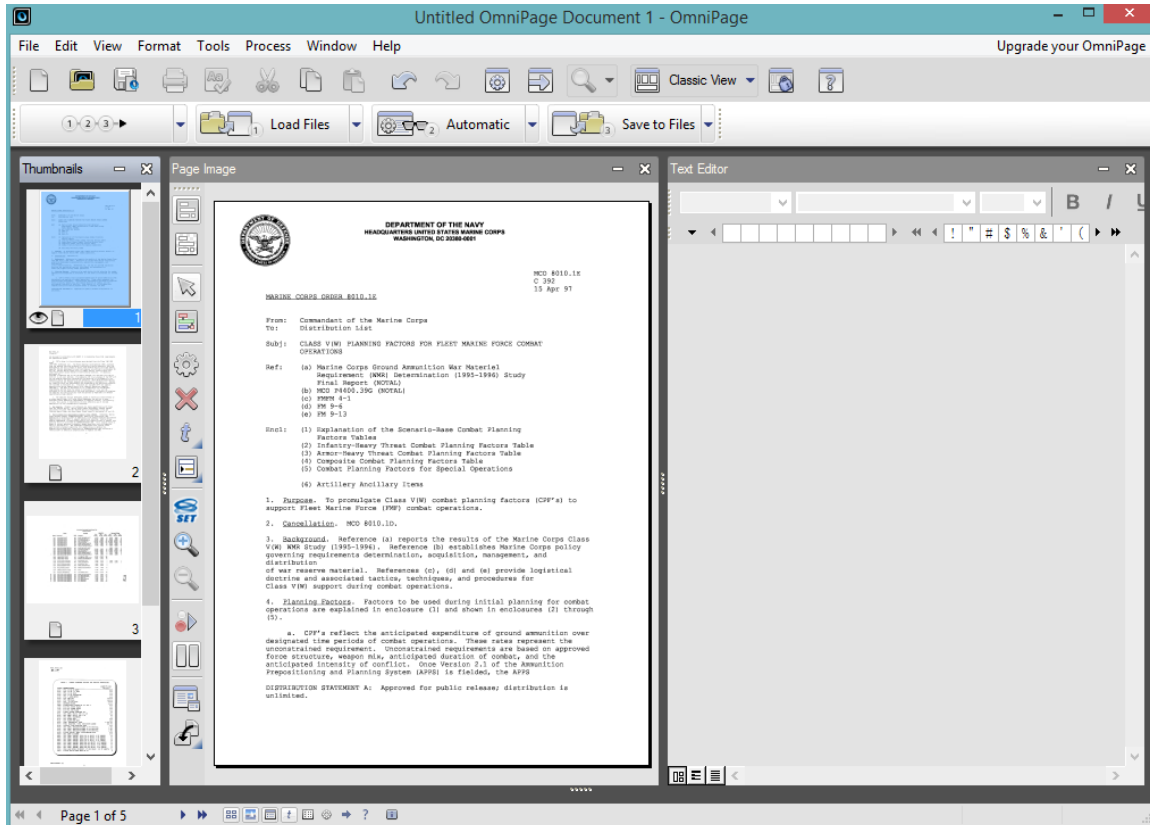


Figure 39. OmniPage® Workspace

The three main frames of the application are a thumbnail screen (left), a page image screen (middle), and a text editor screen (right). Before OCR has been conducted, only the thumbnail screen and page image screen contain information. In order to start processing the document, the user must click a button aptly named “Start Processing.” Once the button is clicked, OCR is performed on the input documents, the text editor screen is populated to show the output, and a proofreading screen appears to walkthrough the document. Figure 40 represents the state of the application once the “start processing”

button has been clicked. For clarity, Figure 41 illustrates the proofreading screen separately.

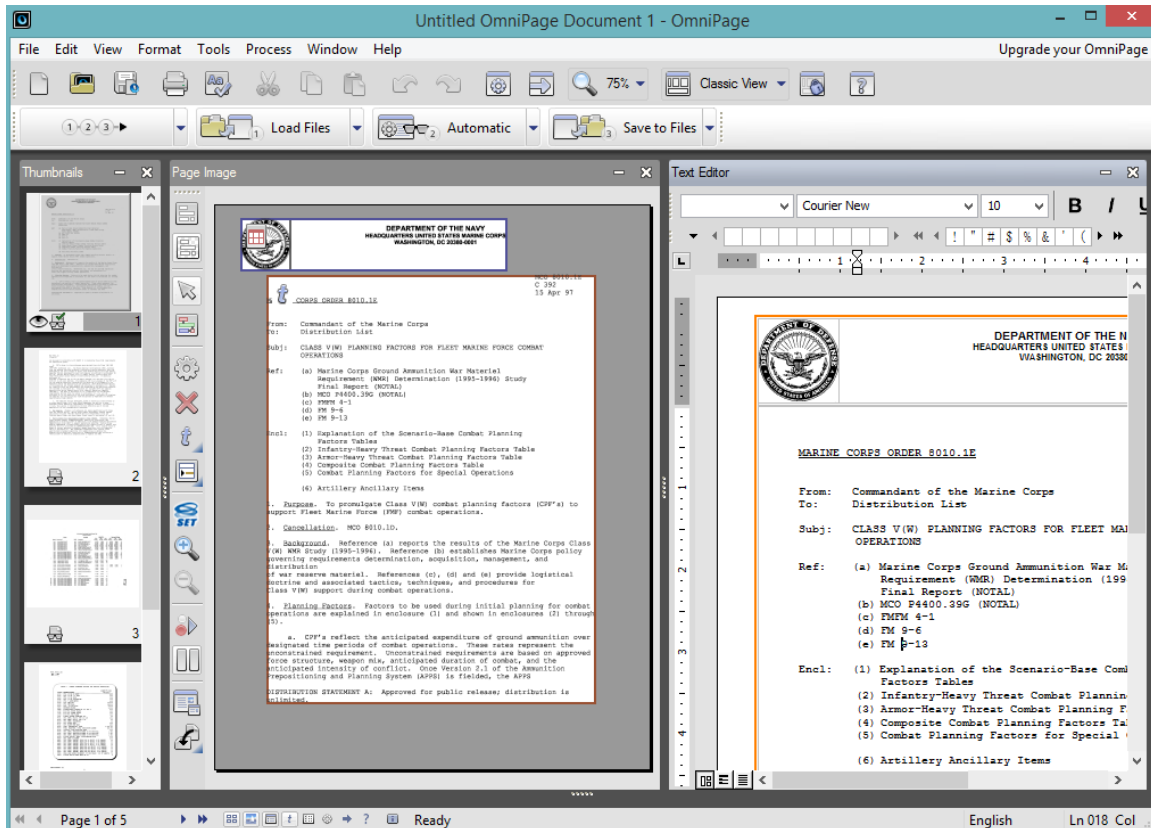


Figure 40. OmniPage® Workspace post-OCR

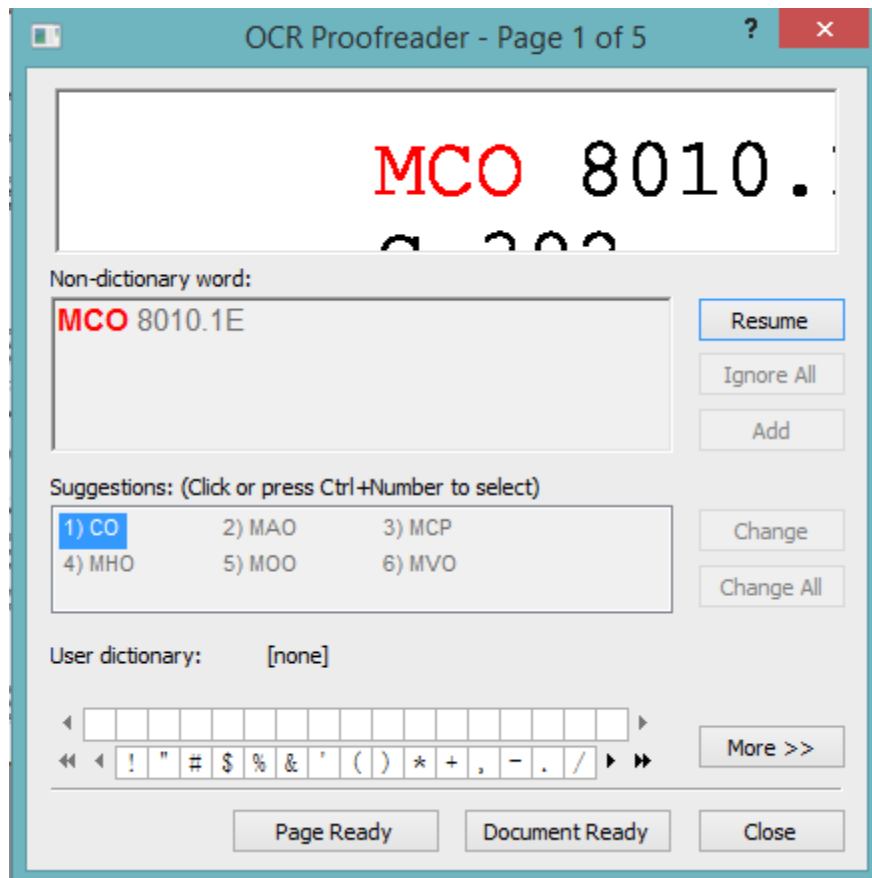


Figure 41. OmniPage® Proofreader

The proofreader screen allows the user to walkthrough the document to verify two cases: spelling and suspected inaccuracy. If the program encounters a word not in its dictionary and is considered a misspelling, “MCO” for example, it allows the user to change it or add it to the dictionary. Adding the word to the dictionary eliminates the need to correct the word later in the document or in future documents. If the program encounters a word that it suspects is an inaccurate transcription, it will prompt the user to enter the correct entry. Both of these cases are handled at the same time on a line-by-line basis. To aid the user, the application shows the original entry in the top box and allows the user to retype the information in the middle box. Once the user has reviewed and corrected the OCR output, he must click the button “save to files.” This opens a dialog box and prompts the user to enter a filename and desired output file type. Figures 42 through 45 illustrate the OCR output produced by OmniPage®.



DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON, DC 20380-0001

MCO 8010.1E
C 392
15 Apr 97

MARINE CORPS ORDER 8010.1E

From: Commandant of the Marine Corps
To: Distribution List

Subj: CLASS V(W) PLANNING FACTORS FOR FLEET MARINE FORCE COMBAT
OPERATIONS

Ref: (a) Marine Corps Ground Ammunition War Materiel
Requirement (WMR) Determination (1995-1996) Study
Final Report (NOTAL)
(b) MCO P4400.39G (NOTAL)
(c) FMFM 4-1
(d) FM 9-6
(e) FM 9-13

Encl: (1) Explanation of the Scenario-Base Combat Planning
Factors Tables
(2) Infantry-Heavy Threat Combat Planning Factors Table
(3) Armor-Heavy Threat Combat Planning Factors Table
(4) Composite Combat Planning Factors Table
(5) Combat Planning Factors for Special Operations
(6) Artillery Ancillary Items

1. Purpose. To promulgate Class V(W) combat planning factors (CPF's) to support Fleet Marine Force (FMF) combat operations.

2. Cancellation. MCO 8010.1D.

3. Background. Reference (a) reports the results of the Marine Corps Class V(W) WMR Study (1995-1996). Reference (b) establishes Marine Corps policy governing requirements determination, acquisition, management, and distribution of war reserve materiel. References (c), (d) and (e) provide logistical doctrine and associated tactics, techniques, and procedures for Class V(W) support during combat operations.

4. Planning Factors. Factors to be used during initial planning for combat operations are explained in enclosure (1) and shown in enclosures (2) through (5).

a. CPF's reflect the anticipated expenditure of ground ammunition over designated time periods of combat operations. These rates represent the unconstrained requirement. Unconstrained requirements are based on approved force structure, weapon mix, anticipated duration of combat, and the anticipated intensity of conflict. Once Version 2.1 of the Ammunition Prepositioning and Planning System (APPS) is fielded, the APPS

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

Figure 42. Post-OCR Results of EX-1 using OmniPage® (after [2])

Infantry-Heavy Threat Combat Planning Factors Table									
Weapon ID Sequence									
Weapon		Ammunition		GCE RATES			CHARGE		
WeaponID	Nomenclature	Nomenclature		Daily ASSAULT	Daily SUSTAIN	Basic Allowance	Daily ASSAULT	Basic Allowance	
B0471	SQUAD DEMOLITION SET	M032	CHARGE, DEMO BLOCK 1 LB TNT	15.00753	3.39805	48	15.00753	3.39805	15
B0471	SQUAD DEMOLITION SET	M130	CAP, BLASTING ELECTRIC	18.01945	4.00000	180	18.01945	2.03084	150
B0471	SQUAD DEMOLITION SET	M131	CAP, BLASTING NON-ELECTRIC	80.00000	36.00000	260	60.00000	36.00000	180
B0471	SQUAD DEMOLITION SET	M456	CORD, DETONATING PETN	1401.30488	340.45345	1500	1401.30488	340.45345	1500
B0471	SQUAD DEMOLITION SET	M670	FUZE, BLASTING TIME	500.00000	216.00000	3000	360.00000	216.00000	1500
B0471	SQUAD DEMOLITION SET	M757	CHARGE, ASSEMBLY DEMOLITION	15.00753	3.39805	50	15.00753	3.39805	50
B0471	SQUAD DEMOLITION SET	M788	IGNITER, TIME FUSE BLASTING	90.00000	54.00000	390	60.00000	36.00000	180
B0471	SQUAD DEMOLITION SET	M123	FIRING DEV, DEMO MULTIPURPOSE	0.47873	0.30030	21	0.47873	0.30030	12
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M032	CHARGE, DEMO BLOCK 1 LB TNT				2.66034	0.18844	5
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M131	CAP, BLASTING NON-ELECTRIC	20.00000	5.00000	20	20.00000	5.00000	80
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M456	CORD, DETONATING PETN	211.55965	48.93008	500	211.55965	48.93008	500
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M670	FUZE, BLASTING TIME	120.00000	30.00000	450	120.00000	30.00000	500
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M757	CHARGE, ASSEMBLY DEMOLITION	2.11527	0.48808	4	2.11527	0.48808	4
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M760	IGNITER, TIME FUSE BLASTING	30.00000	7.00000	30	30.00000	7.00000	50
B0472	DEMOLITION EQUIPMENT INDIVIDUAL	M103	FIRING DEV, DEMO MULTIPURPOSE	1.05755	0.48808	12	1.05755	0.48808	4
B0589	COMBAT MOBILITY VEHICLE	A131	CTG, 7.62MM 4 & 1 LINKED	29.18648	2044456	1000			
B0589	COMBAT MOBILITY VEHICLE	B542	CTG, 40MM HEDP LINKED FOR MK19	1594271	11.08704	884			
B0589	COMBAT MOBILITY VEHICLE	B826	GRENADE, LAUNCHER SMOKE IR	278951	1.81975	36			
B1298	LINE CHG LAUNCH KIT TRLR MTD	J143	ROCKET MOTOR, 5 INCH	0.70085	0.13273	2	0.70085	0.13273	2
B1298	LINE CHG LAUNCH KIT TRLR MTD	M913	CHARGE, DEMO LINEAR HE	0.70085	0.13273	2	0.70085	0.13273	2
B1315	LINE CHG LAUNCHER F/ AAVP7A1	J143	ROCKET MOTOR, 5 INCH	2.36044	0.47000	3			
B1315	LINE CHG LAUNCHER F/ AAVP7A1	M125	CHARGE, DEMO LINEAR HE LVT	2.36044	0.47000	3			
B0150	BRIDGE ARMORED LAUNCHER	B826	GRENADE, LAUNCHER SMOKE IR	3.19475	1.30320	24			
B0207	COMMAND LAUNCH UNIT, JAVELIN	JAVL	JAVELIN	0.21895	0.04515	3			
B0311	DESIGNATED MARKSMAN RIFLE	AM 1	CTG, 7.62MM BALL MATCH	8.67150	21.03178	300			

Figure 43. Post-OCR Results of EX-2 using OmniPage® (after [2])

TABLE 2. COMBAT PLANNING FACTORS FOR SPECIAL OPERATIONS

<u>DODIC NOMENCLATURE</u>	<u>QUANTITY PER MEU(SOC)</u>
A011 CTG, 12 GA 00 BUCK	480
A014 CTG, 12 GA 7.5 SHOT	480
A023 CTG, 12 GA SLUG	250
A024 CTG, 12 GA LOCKBUSTER	300
A136 CTG, 7.62MM MATCH	460
A260 CTG, 9MM JHP	18,000
A363 CTG, 9MM BALL	18,000
A475 CTG, .45 CAL BALL	6,000
AX14 SHOTGUN PRIMER	1,000
DWBS DIVERSIONARY CHARGE MK 141 MOD 0	500
L302 SIG CTG, WHITE FLARE	350
L304 SIG CTG, GREEN FLARE	350
L328 SIG CTG, RED FLARE	350
LX11 SIGNAL ROCKET LAUNCHER KIT	50
M031 CHG, DEMO, BLOCK, TNT 1/2 LB	192
M032 CHG, DEMO, BLOCK, TNT 1 LB	96
M039 CHG, DEMO, CRATERING	20
M130 CAP, BLAST ELEC	288
M131 CAP, BLAST NON-ELEC	480
M456 CORD, DETONATING, REIN	2,000 FT
M670 FUZE, BLASTING, TIME, EXPLOSIVE LOADED	800 FT
M766 IGNITER, TIME BLASTING FUZE	300
M960 CHG, DEMO, EXPLOSIVE SHEET 38 FT PER ROLL	2 RO
M981 CHG, DEMO, EXPLOSIVE SHEET 25 FT PER ROLL	2 RO
M982 CHG, DEMO, EXPLOSIVE SHEET 19 FT PER ROLL	2 RO
ML03 FIRING DEVICE, DEMO, MULTIPURPOSE M124	112
MM30 CHG, DEMO 20 GRAMS	200
MM41 CHG, DEMO, SHAPED, FLEX LIN 30 GR/FT, 6 FT LENGTH	12
MM42 CHG, DEMO, SHAPED, FLEX LIN 40 GR/FT, 6 FT LENGTH	32
MM43 CHG, DEMO, SHAPED, FLEX LIN 60 GR/FT, 6 FT LENGTH	12
MM44 CHG, DEMO, SHAPED, FLEX LIN 75 GR/FT, 6 FT LENGTH	28
MM45 CHG, DEMO, SHAPED, FLEX LIN 100 GR/FT, 6 FT LENGTH	12
MM46 CHG, DEMO, SHAPED, FLEX LIN 225 GR/FT, 6 FT LENGTH	12
MM47 CHG, DEMO, SHAPED, FLEX LIN 400 GR/FT, 6 FT LENGTH	12
MM48 CHG, DEMO, SHAPED, FLEX LIN 600 GR/FT, 6 FT LENGTH	12
MM56 DUAL LEAD NONEL PRIMADET, 175 MS DELAY, 100 FT LENGTH 75	
MN14 FIRING DEVICE HAND HELD MK 54	5

ENCLOSURE (5)

Figure 44. Post-OCR Results of EX-3 using OmniPage® (after [2])

ARTILLERY ANCILLARY ITEMS

APR 15 1997

PROJECTILE		ANCILLARY ITEM		
DODIC	Nomenclature	DODIC	Nomenclature	Multplier
D501	PROJ 155MM, ADAM-L, M692	D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		0540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762 (N285, FZ, MTSQ, M577 may substitute)	1.05
		N523	PRIMER, PERCUSSION, M82	1.1
D502	PROJ 155MM, ADAM-S, M731	D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		0541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762 (N285, FZ, MTSQ, M577 may substitute)	1.05
		N523	PRIMER, PERCUSSION, M82	1.1
0505	PROJ 155MM, ILLUM, M485A2	0533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		0540	CHG PROP 155MM, GREEN BAG, M3	0.15
		0541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762 (N285, FZ, MTSQ, M577 may substitute) (N248, FZ, MT, M565 may substitute)	1.05
		N523	PRIMER, PERCUSSION, M82	1.1
D510	PROJ 155MM, COPPERHEAD, M712	0533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		0540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N523	PRIMER, PERCUSSION, M82	1.1
D514	PROJ 155MM, RAAM-L	D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762 (N285, FZ, MTSQ, M577 may substitute)	1.05
		N523	PRIMER, PERCUSSION, M82	1.1
D515	PROJ 155MM, RAAM-S	D533	CHG PROP 155MM, RB/WB, M119A1/	0.27
		D540	CHG PROP 155MM, GREEN BAG, M3	0.15
		D541	CHG PROP 155MM, WHITE BAG, M4	0.68
		N289	FUZE, ET, M762 (N285, FZ, MTSQ, M577 may substitute)	1.05
		N523	PRIMER, PERCUSSION, M82	1.1
D528	PROJ 155MM, SMOKE, WP, M825	0532	CHG PROP 155MM, RED BAG, M203	0.2
		D533	CHG PROP 155MM, RB/WB, M119A1/	0.2
		D541	CHG PROP 155MM, WHITE BAG, M4	0.7
		N289	FUZE, ET, M762 (N285, FZ, MTSQ, M577 may substitute)	1.05
		N523	PRIMER, PERCUSSION, M82	1.1

ENCLOSURE (6)

Figure 45. Post-OCR Results of EX-4 using OmniPage® (after [2])

Table 5 illustrates a summary of the accuracy rate and important findings for OmniPage®.

Page	Total Errors	Word Errors	Number Errors	Cause / Findings
EX-1	0	0	0	<ul style="list-style-type: none"> Recreated near-perfectly.
EX-2	14	7	7	<ul style="list-style-type: none"> Text converted from portrait view to landscape. Data placed in a table data structure.
EX-3	4	3	1	<ul style="list-style-type: none"> Data placed in a table data structure.
EX-4	5	2	3	<ul style="list-style-type: none"> Near-perfect table recreation.
Totals	23	12	11	

Table 5. OmniPage® Results and Findings

In general, OmniPage® accurately transcribed each input file. In most cases, the application created fully modifiable outputs that were near-duplicates of the input files. While the program did encounter errors, no specific trends appeared. When an error was encountered, it was corrected with the proofreader. The application was able to detect and represent data in different views—landscape and portrait, and also created very accurate and defined tables. Based on the findings for this application, the following scores were given:

- Accuracy: 3
- Consistency: 3
- Speed: 3
- Ease of Use: 1
- Functionality: 3

OmniPage® had the highest accuracy rate of all the applications. It consistently produced the same results. The applications OCR speed and proofreader allow the user to quickly review and correct the document. Using and understanding the basic functionality of the program can be accomplished in approximately one hour. Understanding the advanced functionality of the program can be accomplished in 2-3 hours. The application has fewer input file types when compared to OneNote®, however, it has the most output types of all three applications.

6. OCR Summary

Table 6 illustrates a summary of the scores given to all the applications.

	Accuracy	Consistency	Speed	Ease of Use	Functionality	Totals
Onlineocr.net	2	3	1	3	1	10
OneNote®	1	1	1	2	2	7
OmniPage®	3	3	3	1	3	13

Table 6. OCR Summary Scores

In general, the OmniPage® software out-performed the other applications and received the highest score. Not only did it have the highest accuracy rate, it provided the most functionality—spellcheck, native text editor, and the most output formats. Based on these findings, OmniPage® was used to create the text files that were later used by the extraction programs.

The second-best application was the open-source application. Although the application proved to be accurate, intelligent, and quickly learnable, it is limited by its page-per-hour restriction and number of input and output formats.

OneNote® was least-favored because it produced highly inaccurate and inconsistent results. While the program provides spellcheck functionality and a vast array of input formats, it has limited output options and requires heavy user-involvement to correct the OCR outputs.

B. PROGRAM DEMONSTRATION

After the OCR comparison was conducted, two programs were created to extract the text-based consumption data text from the text files produced by OmniPage®. The goal of the first program was to automate the process of data extraction by using pre-defined decision-making logic. While the program strives for automation during the text extraction phase, user interaction is necessary at the end to verify the outputs. The goal of

the second program was to involve the user in every decision. Since it had no pre-defined logic statements, the responsibility for deciding whether a consumption data element was true and accurate was placed on the user. These programs were created with one assumption: the input document was in an acceptable format for the application and free of errors. Thus, the programs have been given “perfect inputs” which allows the testing to focus solely on data extraction.

The automated program was tested first in a two-phase process. During the first phase, each page was examined separately to determine what unique characteristics existed to distinguish desired elements from superfluous information. Once the unique characteristics (if any) had been identified, the program was written to detect them and successfully extract their contents. Some of the input documents followed regulated correspondence procedures, allowing the leveraging of some of their suitable characteristics. During the second phase, all five of the input documents were placed into one document for the application to process as a whole. This tested the ability of the automated application to work as designed. After the automated application was tested, the walkthrough application was designed and tested as two versions: line-by-line and page-by-page.

1. Automated Program

a. File Input and Closing

In order to begin extracting consumption data out of the input files, the program first opens an input file, reads each line into a list, and then closes the input file. While this part of the program produces no output, it handles necessary application overhead to start the process. Figure 46 illustrates the coding of the file open and close function.

```

# This module handles opening of the input file,
# reading in of inputs, and closing of the file.

rawData = []                # Holds the original lines
lineCount = 0
rawDataSize = 0

def fileOpenRoutine(fileName):
    global lineCount
    global rawDataSize
    lineCount = 0
    rawDataSize = 0
    file = open(fileName, 'r')    # Open the file
    for line in file:
        line = line.strip()      # Strip white space
        if (line != ""):        # Remove blank lines
            rawData.append(line) # Add it to the list
            lineCount = lineCount + 1
            rawDataSize = rawDataSize + 1
    file.close()                # Close the input file

#----- Main -----#

print ("Input filename to analyze: ")

fileName = input()

fileOpenRoutine(fileName)

```

Figure 46. Automated Program: File Open and Close

The program begins by asking the user to input a filename. Once the input has been given, the program calls the first function: *fileOpenRoutine(fileName)*. This function takes the name of a file entered by the end user, opens it, reads in each line from the file, records the number of lines, and closes the file. To aid later data extraction and readability, white space is stripped off the beginning and end of each sentence and blank lines are removed.

b. Detect and Extract Document Information

Now that the input file has been read, the program begins by identifying and storing the important identifying information of each document: name, date of publish, subject, etc. Figure 47 illustrates the coding of this function.

```
identifierInfo = []          # Identifier Information
counter = 0

def handleMarineCorpsOrderIdentifyingInformation(rawDataList):
    counter = 9
    subj = ""
    identifierInfo.append(rawData[6]) # Line 6 should be Marine Corps Order and #
    identifierInfo.append(rawData[5]) # Line 5 should be the date

    while "Ref" not in rawData[counter]: # Subj starts on line 9 and ends when it finds Ref
        if (subj == ""):
            subj = rawData[counter]
        else:
            subj = subj + " " + rawData[counter] # Process a multi-line subject
            counter = counter + 1

    identifierInfo.append(subj)

#----- Main -----#

if "MCO" in rawData[3]:      # We look for the indication of a Marine Corps Order
    handleMarineCorpsOrderIdentifyingInformation(rawData)
```

Figure 47. Automated Program: Detect and Extract Document Info

This part of the program begins with a logic test. “if ‘MCO’ in rawData[3] ... *handleMarineCorpsOrderIdentifyingInformation(rawData)*” conducts a logical test to see if the string “MCO” is present in the third line of the document. Due to the template-based nature of the document, the test is passed, and the program executes the function to handle a MCO: *handleMarineCorpsIdentifyingInformation(rawDataList)*. It is important to note that the input document was written in 1997 and may not represent the format of a current MCO. Thus, it is important for a final and fully-implemented application to follow the most current standards and policies.

The function *handleMarineCorpsOrderIdentifyingInformation(rawDataList)* handles the identification of the document name, date it was published, and subject of the document. Since a MCO follows standard correspondence procedures and the program was given perfect input, these data fields were extracted out of the document by using their exact line numbers. For example, MARINE CORPS ORDER 8010.1E resides on line six, the date of the document resides on line five, and the subject of the document begins on line nine and ends when the first occurrence of “Ref:” is encountered. It is common for the subject line to span several lines, requiring the program to link (concatenate) the lines together in order to accurately present the subject field. Testing for “Ref:” also represents one of the first major problems with implementing an automated program. Testing for the presence of this exact string had to be done in order to stop the program from entering an endless loop. Since the program was given perfect inputs, this was not a problem. However, if the user who verifies the OCR output makes an error and allows a different string such as “ref:” to go through, it would cause this particular program to crash.

While this function handles the detection and extraction of the identifying information for the front page of a MCO, it can be used as a template function to handle other documents: field manuals, technical manuals, etc. Figure 48 illustrates the output of this function once the program successfully detected and extracted the document’s identifying information.

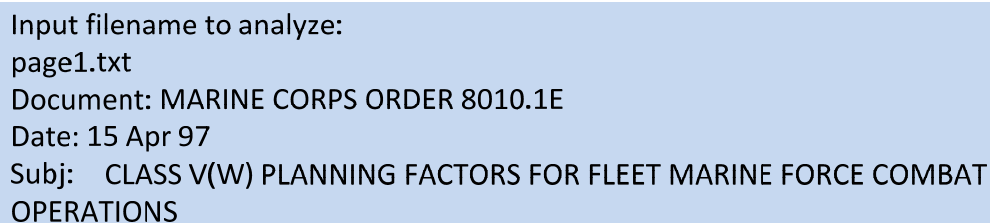
A screenshot of a program's output, enclosed in a light blue rectangular box with a black border. The text is as follows:
Input filename to analyze:
page1.txt
Document: MARINE CORPS ORDER 8010.1E
Date: 15 Apr 97
Subj: CLASS V(W) PLANNING FACTORS FOR FLEET MARINE FORCE COMBAT
OPERATIONS
The text is left-aligned and uses a monospaced font.

Figure 48. Detect and Extract Document Info Output

c. Detect and Extract Table Information

This part of the program is responsible for detecting tables in the documents and extracting their contents. Figure 49 illustrates the coding of this part of the program (v1.0), which tests for the presence of the “Infantry-Heavy Threat Combat Planning Factors Table.” In the example given, the function *handleDataStructures(rawData)* is called by the program after the file has been read and the document’s identifying information has been recorded. By studying the input document, we know that the string, “Infantry-Heavy Threat Combat Planning Factors Table,” represents a table inside the document that contains lines of consumption data. In order to begin data extraction from the table, a logical test is conducted first to see if the table is present in the document: “if ‘Infantry-Heavy Threat Combat Planning Factors Table’ in rawData[counter]” is tested on each line from the input document. After the program detects the table, it reads lines of data into an intermediate data structure, “tableData,” until the next occurrence of the word “Table” is encountered. This had to be done for the same reason for which the occurrence of “Ref:” was tested. Due to the current format of the input document, no distinguishable landmarks existed for the program to stop. The only way to get the program to stop was by testing for the presence of a new table. Again, should the word table not exist or be spelled incorrectly; the program would enter an endless loop, requiring user intervention.

By reading the table contents into the secondary data structure “tableData” we can further isolate the inputs for extraction. Another counter, “tmpCounter,” is used to prevent the main program counter, which controls position, from being adjusted, saving the correct position in the main program until the secondary extraction process has completed. Further leveraging known information from the input document, we know that the DODIC has a length of five characters. Thus, the logical test “if(len(tableData[tmpCounter]) == 5)” is used to indicate when a new line of data is encountered. Once a new line of data is encountered, the previous line of data, “dataLine,” is printed to the screen. Note: some of the logic tests created for this program may cause false positives. For example, if a consumption data element is five characters and not a DODIC, the program would attempt to extract the data based on a false

positive. However, using these tests became necessary because no other distinguishable tests could be created based on the format of the input document. Thus, in order to reduce complexity of the code and minimize false positives, new standards for the input documents may be needed. Furthermore, while the logic test checks for the presence of the “Infantry-Heavy Threat Combat Planning Factors Table,” testing can be done on other inputs. For example, placing “Infantry-Heavy Threat Combat Planning Factors Table,” “Armor-Heavy Threat Combat Planning Factors Table,” and the “Composite Combat Planning Factors Table” in a list data structure could be used to repetitively test for table existence. By using this approach, a “consumption data dictionary” that contains known occurrences of table names could be created. Over time, this dictionary could track all known occurrences of tables and provide template-based formatting for their extraction. For example, if the program were to recognize and detect the presence of an “ammunition table” and have a pre-defined understanding that this table was 10 lines of data, the program could locate the table and extract 10 lines of data. This would help alleviate some of the hard-coded complexities in previous examples. Figure 50 illustrates the output of this function (v1.0).

```

def handleDataStructures(rawData):
    global rawDataSize
    counter = 0
    tmpCounter = 0
    tableData = []
    tableDataSize = 0
    dataLine = ""
    while (counter < rawDataSize):
        if "Infantry-Heavy Threat Combat Planning Factors Table" in rawData[counter]:
            counter = counter + 18 # Move past table header and column headers
            while "Table" not in rawData[counter]:
                tableData.append(rawData[counter])
                tableDataSize = tableDataSize + 1
                counter = counter + 1
            if (counter == rawDataSize): # There are no more inputs, prevent
                break # reaching out of bounds

        while (tmpCounter < tableDataSize): # Construct each line of data
            if (dataLine == ""):
                dataLine = tableData[tmpCounter]
                tmpCounter = tmpCounter + 1

                while (True):
                    if (tmpCounter >= tableDataSize):
                        break
                    dataLine = dataLine + " " + tableData[tmpCounter]
                    tmpCounter = tmpCounter + 1
                    if (tmpCounter >= tableDataSize):
                        break
                    if (len(tableData[tmpCounter]) == 5):
                        print (dataLine)
                        dataLine = ""
                        tmpCounter = tmpCounter - 1
                        break
                tmpCounter = tmpCounter + 1
            counter = counter + 1

```

Figure 49. Automated Program: Handle Data Structures (v1.0)


```

Input filename to analyze:
page11.txt
Found Table:
Infantry-Heavy Threat Combat Planning Factors Table
Known column headers for this table:
      Weapon      Ammunition      GCE RATES      Other than GCE Rates
Weapon ID  Nomenclature DODIC  Nomenclature  Daily  Daily  Basic  Daily  Daily  Basic
              ASSAULT SUSTAIN Allowance ASSAULT SUSTAIN Allowance
B0471 SQUAD DEMOLITION SET M032 CHARGE, DEMO BLOCK 1 LB TNT 15.00753 3.39605 48 15.00753
B0471 SQUAD DEMOLITION SET M130 CAP, BLASTING ELECTRIC 18.01945 4.00000 150 18.01945 2.03084
B0471 SQUAD DEMOLITION SET M131 CAP, BLASTING NON-ELECTRIC 60.00000 36.00000 260 60.00000
B0471 SQUAD DEMOLITION SET M456 CORD, DETONATING PETN 1401.30498 340.45345 1500
B0471 SQUAD DEMOLITION SET M670 FUZE, BLASTING TIME 500.00000 216.00000 3000 380.00000
B0471 SQUAD DEMOLITION SET M757 CHARGE, ASSEMBLY DEMOLITION 15.00753 3.39605 50 15.00753
...

```

Figure 50. Handle Data Structures Output (v1.0)

To illustrate how new policy standards can help reduce the amount of coding and overall complexity of the program, two landmarks were inserted before and after the table. The phrases “Begin Table” and “End Table” were placed in the input document as wrappers around the table. The program was then modified to detect these phrases and conduct data extraction. These changes are illustrated in Figure 51.

```

def handleDataStructures(rawData):
    global rawDataSize
    counter = 0
    tableData = []
    tableDataSize = 0
    while (counter < rawDataSize):
        if "Begin Table" in rawData[counter]:
            while "End Table" not in rawData[counter]:
                tableData.append(rawData[counter])
                tableDataSize = tableDataSize + 1
                counter = counter + 1

```

Figure 51. Automated Program: Handle Data Structures (v2.0)

By placing these phrases into the input file, we were able to drastically reduce the complexity of the program. Rather than creating programs that must handle very specific tests such as detecting the string “Infantry-Heavy Threat Combat Planning Factors

Table,” refinement of the input allows more general usage, regardless of table name and input document type.

d. File Output

This part of the program focuses on placing the consumption outputs into a file. While this program focuses on output to a file, the output produced can be interpreted as a (key, value) pair. This key value would take the following form: a string key consisting of the documents identifying information and a list value that would consist of each line of consumption data. Thus, it would be (string identifyingInformation, list consumptionDataLines). Although this program concentrates on writing outputs to a file at the end of the document’s processing, the outputs could be written to a file as the program works through each consumption table. For example, when a table is encountered, the entire table is written to the file, the data structure that holds the table information is then cleared, and the process is free to move to the next table, allowing repetitive use of the intermediate data structure. Figure 52 illustrates the coding of this function.

```
def writeToFile(fileName, table):  
  
    outputFile = open (fileName, 'w')  
  
    for element in table:  
        outputFile.write(element + "\n")  
  
writeToFile("outputs.txt", table)
```

Figure 52. Automated Program: File Output

While this function produces no visual output, it places all of the contents in the provided table into the provided filename. The contents of this table would consist of all the consumption elements extracted from the input document.

e. Phase Two Test

In order to test the automated program for phase two, all of the input files were consolidated into one file named “consolidatedinputs.txt.” The program was slightly modified to search for all occurrences of the tables inside of the consolidated input file. Again, it became necessary to create unnecessary tests based on false-positives caused by the format of the input document. The exact string “Infantry-Heavy Threat Combat Planning Factors” was found on the first page of the input document in the enclosure section. The program began to extract data from this point, all of which was incorrect. “If ((counter > 90) and (“Infantry-Heavy Threat” in rawData[counter]))” represents the logic test that had to be created in order to access the table at the correct position. This could have been avoided by giving the program only the sections of the document that contained the tables. However, this requires more user interaction. Additionally, the problem of determining when a table had ended presented itself again. In order to stop extracting data for a particular table, the program had to check for the presence of the next table. Using the “Begin Table” and “End Table” changes, as suggested earlier, would have prevented us from having to create these unnecessary tests. Should these tables not exist or their names be misspelled, this program would enter an endless loop. Figure 53 illustrates a program that was capable of extracting consumption data elements from all the tables in the consolidated input file.

```

table1 = []
table2 = []
table3 = []

while (counter < rawDataSize):
    if ((counter > 90) and ("Infantry-Heavy Threat" in rawData[counter])):
        table1.append(rawData[counter])
        counter = counter + 1

        while "COMBAT PLANNING FACTORS FOR SPECIAL OPERATIONS" not in rawData[counter]:
            table1.append(rawData[counter])
            counter = counter + 1

    if "COMBAT PLANNING FACTORS FOR SPECIAL OPERATIONS" in rawData[counter]:
        table2.append(rawData[counter])
        counter = counter + 1

        while "ARTILLERY ANCILLARY ITEMS" not in rawData[counter]:
            table2.append(rawData[counter])
            counter = counter + 1

    if "ARTILLERY ANCILLARY ITEMS" in rawData[counter]:
        table3.append(rawData[counter])
        counter = counter + 1

        while len(rawData[counter]) > 0:
            table3.append(rawData[counter])
            counter = counter + 1

        if counter >= rawDataSize:
            break

    counter = counter + 1

```

Figure 53. Automated Program: Handle Consolidated Input File

f. Automated Program Summary

The automated program takes only one user input to begin processing—a filename. Once the filename has been entered, the program executes automated analysis of the file using pre-built logic tests. Once the program has finished its analysis, the user must verify each data element before it is written to the output file. This part of the program is not illustrated because it is very similar to the program presented in the next section.

While the goal of this program is to achieve automation, it requires extensive testing, logic creation, and still requires user interaction once the program has autonomously extracted out all the possible consumption data elements to review the results. Some of the logic tests used by this program are highly unnecessary and may ultimately cause the program to fail. For example, the test “If ((counter > 90) and (“Infantry-Heavy Threat” in rawData[counter]))” would be necessary unless user selection of the input data is given or refinement of the input document occurs. While the goal of creating these tests was to increase the accuracy of the output, they increase the complexity and length of the code, require additional processing power, and may cause the program to perform slower or more inefficiently. Additionally, they may only work with very specific inputs. In order to mitigate this problem, refinement of the input document is necessary. By creating unique identifiers such as “Begin Table” and “End Table,” the complexity and length of the program can be reduced while also allowing it to accept a larger variety of inputs.

2. Walkthrough Programs

While the automated program makes use of several functions, this program only uses one. Instead of extracting the document’s identifying information and consumption elements separately, this program allows the user to step through each line of the input document in sequential order, prompting the user to keep the line or disregard it. Thus, the need to separately process identifying information and consumption elements can be done simultaneously since both will come through as lines of input. Using this approach, the input file can be analyzed line-by-line or another increment: paragraph-by-paragraph, page-by-page, etc. Therefore, two approaches using this application are presented: line-by-line and page-by-page. As a reference point, a page is defined as 45 lines of input. The decision to use 45 lines is based on the standard MS Word® document format. A one-page document with one-inch borders can hold 45 lines of information in Times New Roman, 12-pitch font.

a. *Line-by-line Program*

First, the input file is opened and each line of the document is read into a temporary data structure. Afterwards, the user is presented with one line of information at a time and is prompted to verify whether or not it is a data element. If, and only if, the user enters “yes,” the element gets placed into the final output data structure. Once the document has been reviewed and no inputs remain, the data structure that contains all the “yes” responses is then placed into an output file. Figure 54 illustrates the coding of the line-by-line program. Figure 55 illustrates a snapshot of the running application.

```
validInformation = []                # Store the final data elements

def manualWalkthrough(rawDataList):
    for element in rawDataList:
        print ("Is this valid information?")
        print (element)
        response = input()
        if (response == "yes"):
            validInformation.append(element) # If yes, add it to the list

manualWalkthrough(rawData)          # Call the function
```

Figure 54. Line-by-line Program (coding)

```
Is this valid information?
DEPARTMENT OF THE NAVY
yes
Is this valid information?
HEADQUARTERS UNITED STATES MARINE CORPS
yes
Is this valid information?
WASHINGTON, DC 20380-0001
no
Is this valid information?
MCO 8010.1E
...
```

Figure 55. Line-by-line Program (running)

The main strength of this approach is its simplicity. The main weakness of this approach is its lack of functionality. As long as there is only one valid consumption

element per line in the input document, an end-user can quickly and accurately walk through the document. However, if the consumption data element is split apart and spread across multiple lines, the program output may not make sense. In order to correct this, the input document must be further processed or a string concatenation procedure must be created. While the program can quickly walk through each line, the end user may find it faster to view multiple lines of information at once with the ability to select specific lines or ranges. Thus, the page-by-page program offers a potential performance increase.

b. Page-by-page Program

This program is similar to the line-by-line program but aims to speed up the review process. Figure 56 illustrates the coding of this program.

```
def manualWalkthrough(rawDataList):
    counter = 1
    tmpCounter = 0
    while (True):
        print ("Please select the lines of valid information by line #:")
        print ("Separate line #'s with a space - e.g. 1 3 15 31 44")

        while ((counter < 46) and (tmpCounter < rawDataSize)):
            print(str(counter) + "." + " " + rawDataList[tmpCounter])
            tmpCounter = tmpCounter + 1
            counter = counter + 1

        response = input()

        responses = response.split(" ") # Split the inputs by their space

        for element in responses:
            validInformation.append(rawDataList[tmpCounter])

        counter = 1

        if (tmpCounter == rawDataSize):
            print ("Review complete.")
            break

manualWalkthrough(rawData)      # Call the function
```

Figure 56. Page-by-page Program (coding)

The main strength of this approach is its speed. The main weakness is its need for structured and accurate input by the user. The program begins by prompting the user to enter specific line numbers that are deemed to be elements or consumption data. Before the user is able to enter input, 45 lines of data are presented to the user for review. The user must enter structured input (line numbers separated by white space) that corresponds to each correct line. Once the user input has been tied to corresponding lines, the respective lines are transferred to the final output data structure. Once all of the lines have been reviewed, the program terminates. An extension of the program would be the ability to go back through the final output list and modify and review the values as a second layer of precaution. While the goal is to speed up the process, it places more responsibilities and requirements on the end user. Unless failure logic is added, inappropriate responses or errors in the input will cause the program to fail or raise exceptions. Figure 57 illustrates a running instance of the program.

```
Please select the lines of valid information by line #:
Separate line #'s with a space - e.g. 1 3 15 31 44
1. DISTRIBUTION STATEMENT A: Approved for public release; distribution
2. is unlimited.
3. MCO 8010.1E
4. 15 Apr 97
5. can be used in conjunction with MAGTF II to determine Class V(W)
requirements
6. for operational plans.
...
1 2 3 4
Output:
1. DISTRIBUTION STATEMENT A: Approved for public release; distribution
2. is unlimited.
3. MCO 8010.1E
4. 15 Apr 97
```

Figure 57. Page-by-page Program (running)

3. Program Summary

The first program concentrates on automated analysis with user interaction at the end of the automated data processing. While this program strives to achieve automation,

additional consumption document analysis and logic test creation is required. The creation of complex logic tests can be mitigated by further refinement of the input document. The second program concentrates on walkthrough analysis using two approaches: line-by-line and page-by-page. While the line-by-line program offers simplicity in its current state, it lacks functionality and may be slower than the second approach. Using the page-by-page approach, speed is offered at the loss of simplicity, as well as added reliance on user correctness, allowing 45 pages of information to be reviewed at a time. Since consumption data may be “buried” at the end of a consumption document, this approach allows the end-user to quickly reach their desired position in the document. Regardless of which program is used, the emphasis remains on correcting and ensuring that the input document is free of errors and is in the most optimal format. This can be accomplished through multiple layers of internal review and the establishment of new standards that would govern all future consumption documents.

V. CONCLUSIONS

A. LESSONS LEARNED

1. Choosing OCR Application

Chapter II discussed two OCR approaches: free-form and template-based recognition. Due to the nature of the consumption documents reviewed by this thesis, free-form analysis was chosen over template-based. Although template-based OCR can be faster than free-form, no feasible template opportunities presented themselves. Additionally, template-based OCR requires a new template be created for every instance of a table. Thus, free-form OCR is the most-preferred method for the current state of consumption documents. Should consumption data be presented in a template format in future standards or documents, template-based OCR may present itself as an opportunity.

Chapter IV compared three OCR applications: an open-source, online application, Microsoft OneNote®, and Nuance OmniPage®. Of the three applications, the OmniPage® software offered the most-reliable functionality and the highest accuracy rate and is therefore recommended over the other applications.

OmniPage® allowed for a simultaneous conversion and correction process, removing the requirement to separately correct the input document in another text editor after it had been converted. The open-source, online application was more accurate and more robust than OneNote®. Although the open-source application claims to be free, the number of pages that can undergo OCR are limited unless additional pages are purchased. Thus, while it may not be feasible to use the online application, the OCR conversion process could be out-sourced at first until native OCR capability is acquired or created. OneNote® proved to be unreliable, creating numerous spelling errors while not providing intelligent functionality for creating tables or data structures.

2. Choosing Application Approach

Chapter IV compared two programs created to extract consumption information out of input documents: automated and walkthrough analysis. The walkthrough analysis

program was further subdivided into two approaches: line-by-line and page-by-page. While the automated program was able to autonomously extract consumption elements out of the input document, highly-complicated and restrictive parsing logic had to be created. Creation of this logic requires that the programmer understand and be familiar with the nature of the input document. Additionally, a programmer would need to see an example of every consumption document to ensure that they had correctly defined all the logic statements. Furthermore, creation of these restrictive tests should be minimized to ensure the application could be used in a wide range of environments. While the template-based nature of a consumption document such as a MCO allows the program to accurately extract the documents identifying information, it can also create problems that must be addressed. For example, consumption tables are commonly referenced as enclosures. Thus, logic had to be created to look past these occurrences. Additionally, user interaction must occur at the end of the program to verify that the program correctly interpreted and extracted all the possible data elements.

Since the current state of consumption documents may require extensive logic test creation, the goal of the walkthrough analysis program was to circumvent this requirement. Allowing the user to walk through the input document in a line-by-line or page-by-page basis, the responsibility of verifying consumption data is placed on the end-user instead of the program. While both approaches allow the end-user to walk through the input document, the page-by-page program is preferred since it allows for rapid movement, selection, and verification.

Regardless of which program is used, the need to refine and format the input document remained a central focus throughout the process. First, the input document should be converted to an appropriate input format (.docx, .txt, .PDF) for the extraction program. In this thesis, a text file with a .txt extension was used to keep the input very simple and to allow the native Python libraries to open the files. Once the document has been placed into the appropriate format, it should be reviewed and reformatted (spelling, table creation, etc.) in order to make the document easier to read for the end-user of the program. While many of these steps may be required due to the current state of consumption documents or the system as a whole, future standards and policies can be

dictated and followed to reduce the steps required in this process. Although the program focuses on output to a file, it can be altered slightly to output a (key, value) pair. The decision as to what output to use is based on the storage approach used by the end-user. This part of the problem was left for future analysis and research.

B. RECOMMENDATIONS

This thesis recommends the following:

- Establish a baseline listing and collection of consumption documents in one central location. In order to create a reliable automated application, the program should be aware of all known iterations of a consumption document.
- Convert the input documents using OCR software that primarily focuses on OCR or outsourcing to achieve the desired input format. Although professional-grade software such as OmniPage®, isn't free, it offers the highest accuracy and most functionality at a relatively low cost.
- Refine and utilize a page-by-page walkthrough analysis program to extract and upload the first iteration of consumption data elements, using the baseline listing as a checklist.
- Refine standard policies and correspondence procedures for the representation of consumption data in future documents.
- Once the baseline has been established and refined policies have been created, refine and utilize an automated analysis program.

Due to the current format state of consumption documents and the lack of a centrally - located baseline listing, implementation of an automated program would be ineffective. Thus, the baseline should be created using a walkthrough program and then gradually migrated to a point where an automated program can produce accurate results without unnecessary coding.

C. FUTURE RESEARCH

The following areas represent opportunities for future research and development:

- Although commercial-off-the-shelf OCR technology was compared, a native OCR application could be researched and created. Conducting research in this field would require extensive background in computer vision, text analysis, and application coding and may require more than one thesis to fully-develop the application.

- While a “bare-bones” walkthrough analysis program was given, the creation and refinement of a walkthrough application might be accomplished in the scope of a single thesis.
- Further advancement and creation of an automated program would be best suited for a single thesis. Should this avenue be pursued, it is recommended that the researcher have a background understanding and access to Marine Corps logistics documents and be proficient in application coding.
- While the programs that are presented make use of safe coding practices, they do not focus on security vulnerabilities that may or may not be present. Thus, vulnerability testing could be conducted.
- This thesis addresses how inputs can be placed into a database. It does not illustrate how this data can best be presented to the end user. Thus, data access and representation can be researched from a Human Computer Interaction (HCI) standpoint.

D. SUMMARY

In conclusion, this thesis has strived to provide the best picture for the way forward by conducting background research into OCR and presenting multiple approaches for tackling the analysis phase. The purpose of the examples given, problems identified, and recommendations presented is to support the Marine Corps advance towards automation of logistics consumption data and associated planning, allowing its focus to remain on winning the nation’s wars.

LIST OF REFERENCES

- [1] A. Feickert. (2014, Jan. 9). "Marine Corps drawdown, force structure initiatives, and roles and missions: Background and issues for Congress." [Online]. Available: <http://www.fas.org/sgp/crs/natsec/R43355.pdf>
- [2] P. Van Riper (1997, Apr 15). "Class V(W) planning factors for fleet marine force combat operations." [Online]. Available: <http://www.globalsecurity.org/military/library/policy/usmc/mco/8010-1/mco8010-1e.pdf>
- [3] J. Webb et al., "MAGTF logistics planning factors study," Decision Engineering, Woodbridge, VA, Contract Number M00264-01-D-0002, Mar. 22, 2006.
- [4] "DoD issuances, the official department of defense website for DoD issuances." [Online]. Available: <http://www.dtic.mil/whs/directives/corres/ins1.html>
- [5] P. Black. "Dictionary." Dictionary of Algorithms and Data Structures [Online]. Available: <http://xlinux.nist.gov/dads/HTML/dictionary.html>
- [6] L. Eikvil. (1993, Dec). "OCR, optical character recognition." [Online]. Available: <http://www.nr.no/~eikvil/OCR.pdf>
- [7] "OCR-free-form vs template." [Online]. Available: <http://www.documation.co.uk/accounts-payable/ocr-free-form-vs-template/>
- [8] "Template." (2014, Aug 6). [Online]. Available: <http://www.merriam-webster.com/dictionary/template>
- [9] "Free online OCR-convert scanned PDF and images to word, JPEG to word." (2014, Aug 6). [Online]. Available: <http://www.onlineocr.net/>

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California